Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FINAL DE GRADO

TÍTULO DEL TFG: Diseño de un panel de simulación de vuelo portátil

TITULACIÓN: Grado en Ingeniería de Aeronavegación

AUTOR: Daniel Rubén Ruiz Cifuentes

DIRECTOR: Ramón Casanella Alonso

FECHA: 30/01/2018

Resumen

Los simuladores de vuelo son herramientas muy útiles que permiten la formación de nuevos pilotos, la mejora de las habilidades de vuelo y la práctica de procedimientos y situaciones de emergencia en un entorno controlado.

Son utilizados por escuelas de aviación, centros de estudios de aeronáutica e incluso, lugares de ocio.

Los simuladores de vuelo son capaces de representar prácticamente cualquier avión real y sus funciones. Además, son elementos muy llamativos visualmente por sus instrumentos, que atraen la atención del público.

En este proyecto se investigará cómo crear simuladores de vuelo, y se propondrá un diseño para poder ser construido en la Universidad Politécnica de Cataluña a petición de su profesorado y Dirección, para poder ser trasladado a eventos de promoción de la escuela.

Para poder realizar el diseño, se trabajará en el entorno de las tarjetas electrónicas IOCards y con su software SIOC, específicos para simulación aérea.

Title: Design of a portable flight simulator panel

Author: Daniel Rubén Ruiz Cifuentes

Director: Ramón Casanella Alonso

Date: 30/01/2018

Overview

Flight simulators are very useful tools, that allows the training of new pilots, the improve of flight skills and the practice of emergency situations in a controlled environment.

They are used in flight schools, aeronautical study centers and even, in leisure places.

Flight simulators are capable to represent practically any real plane and its functions. In addition, they are very eye-catching because its flight instruments on board, that they capture the attention of the public.

In this project it will be investigated how to make flight simulators and I'll propose one design to be built in a future in the school, requested by the faculty that can be moved easily to the different events in which the campus participates.

To do the design, we will work inside the environment of the electronic boards IOCards and its software SIOC, that are specifically built for flight simulation.

A mi madre y Josep porque sin
ellos y su apoyo, no habría
podido alcanzar ninguna meta.

A Jordi Berenguer,
Marcos Quílez,
Ramón Casanella,
David García,
Josep M. Yúfera,
Joshua Tristancho

por su ayuda e interés en los
proyectos de simulación del campus.

A mi profesor Víctor Bosque del
Ciclo Superior de Telecos quien,
con la pasión con las que da sus
clases, me hizo despertar el
interés por la Ciencia.

Y a mi amigo Edgar Cabrera

ÍNDICE

INTRODUCCIÓN

En este proyecto de final de grado se ha estudiado la manera de crear simuladores de vuelo para poder realizar el diseño de un simulador portátil para ser utilizado en la Universidad.

La petición, propuesta por el profesorado del campus y aprobada por Dirección, viene motivada por el antiguo simulador de vuelo de la escuela, el panel de simulación del cual fue construido por mí en 2013 **(Fig I.1)** y la necesidad de poder trasladarlo a eventos de promoción como medio de publicidad de la escuela.

Figura I.1: Simulador actual de la Universidad

El simulador antiguo, aunque es muy llamativo y funcional y ha sido utilizado en multitud de eventos, como en la feria de simulación más grande de España[1]: el Air Sim Meeting; charlas y talleres de la asociación de estudiantes: la AEAC[2], al encontrarse anclado al suelo, no es posible su desmontaje ser movido fácilmente ya que no fue diseñado para ello.

Tener un simulador portátil que pueda ser mostrado a toda clase de público en cualquier lugar, puede ayudar a promocionar la universidad y sus estudios. Los

[1] http://www.cealweb.net/IPBCEAL/index.php?/page/index.html/_/noticias/simulacion-aerea/barcelona-air-sim-meeting-2013-r121
[2] http://www.aeronauticscat.org/

simuladores de vuelo son tan llamativos que se pueden usar como reclamo y, así despertar la curiosidad por nuestra Universidad. Además, tendrá una funcionalidad real simulando paneles de cualquier monomotor o bimotor que se encuentre instalado.

Su uso no se limitará a la promoción. Aprovechando que los simuladores de vuelo son herramientas extremadamente versátiles, el nuevo equipo permitirá la práctica de todo lo aprendido en clases de aeronáutica (procedimientos, cartas de vuelo, física aeronáutica), electrónica (mediante las tarjetas electrónicas que forman el panel) y programación (con la programación de dichas tarjetas), entre otras. Además, dada su facilidad de transporte, a diferencia del actual simulador, será más sencillo que los profesores le den uso, ya que no será necesario trasladar a todos sus alumnos a un aula pequeña, si no, que se podrá trasladar el equipo directamente a la clase y ser mostrado desde cualquier ángulo de manera sencilla.

El simulador portátil cuenta con un presupuesto aprobado de 3000€ y se ha diseñado cumpliendo las necesidades propuestas por la universidad: que fuera fácilmente trasladable, que sea estéticamente llamativo y además robusto y funcional.

En todas las fases de diseño y de su construcción, se ha intentado mantener un carácter artesanal, reduciendo el coste final del proyecto e intentando darle una apariencia única, que destaque allí donde sea mostrado, mediante una cuidada disposición de los instrumentos de vuelo y un panel llamativo de metacrilato con unos detalles finales muy cuidados dando una apariencia profesional.

Además, se ha intentado escoger, en la medida de lo posible, elementos respetuosos con el medio ambiente, como madera en lugar de plásticos, y la compra de componentes que cumplan la normativa de medio ambiente pertinente.

Por todo ello, el nuevo simulador se ha diseñado teniendo en cuenta los siguientes puntos:

- Ha de ser portátil
- Visualmente llamativo
- Funcional
- De coste no superior a 3000€ (ordenador y monitores incluidos)
- Ha de ser robusto para resistir el paso del tiempo y el uso de muchas personas
- No ha de fallar a nivel de programación
- Ha de ser fácil de usar y mantener
- Ha de ser, en la medida de lo posible, ecológico

Para poder realizar con éxito el diseño del panel portátil, se ha tenido que aprender a crear simuladores de vuelo mediante el sistema de tarjetas

electrónicas IOCards[3] y su entorno de programación y programario SIOC[4]. Además, con vistas a la realización de este TFG y la creación del nuevo diseño, ya desde 2013 comencé con el estudio y las pruebas de construcción de simuladores personales y el que realicé para la escuela en ese mismo año.

Tal ha sido la experiencia adquirida durante todo este tiempo y con este TFG, que tras haber grabado para YouTube varios vídeos[5] mostrando mis pruebas de simulación caseras, numerosas personas de España, América del Norte y Sudamérica se han puesto en contacto conmigo para realizar consultas técnicas en este campo, siendo los casos más extremos, un ingeniero eléctrico interesado en montar una empresa de simulación en Colombia, y un policía instructor de la Escuela de Aviación Policial, también, de Colombia[6] [7].

Gracias al estudio del sistema IOCards y SIOC, a los distintos simuladores de vuelo construidos y la experiencia adquirida con ellos, puedo decir que el diseño propuesto en este proyecto será factible para ser construido por mí en el momento que la Universidad disponga de los materiales, aunque también, por experiencia puedo prever en la posibilidad, de que una vez se monte, se tenga que variar alguna línea de programación o realizar alguna modificación al diseño propuesto.

Para organizar el trabajo de manera entendible se englobará el diseño en dos capítulos y anexos:

- El primero será la parte de diseño del hardware, en la que se explicará la forma y la instrumentación del panel, y su conexión mediante las tarjetas IOCards.
- El segundo será su programación y configuración, es decir, la parte de software del proyecto.
- Por último, se incluirá como anexos, la programación y la configuración que se usará en el simulador portátil.

Al no haberse podido construir el simulador portátil a tiempo de la presentación de este TFG a causa de asuntos internos de la Universidad, si fuera necesario presentar un ejemplo o explicar un tema poniendo ejemplos reales, se utilizará el otro simulador de la Escuela, ya que comparten una gran parte del funcionamiento.

---

[3] http://www.opencockpits.com/index.php/es/iocards/item/descripcion?category_id=63
[4] http://www.lekseecon.nl/downloads/How%20to%20SIOC.pdf [Enlace a descarga de PDF]
[5] https://www.youtube.com/watch?v=Y9Wr4vdw7sM&list=PLGltNDsmkqccW5uUgawP5fiRt7DNxgGWD
[6] https://www.youtube.com/watch?v=932cpc1r47U&feature=youtu.be
[7] http://www.semana.com/nacion/multimedia/escuela-de-aviacion-policial-los-guardianes-del-cielo/432755-3

Capítulo I. Diseño de hardware

1.1 Requisitos y características

El diseño del simulador portátil propuesto, como ya se ha explicado en la introducción, deberá cumplir una serie de requisitos. Los dos principales y que determinarán la forma y funcionalidad finales del simulador serán la portabilidad y el presupuesto.

Para solucionar el problema de la portabilidad se ha diseñado un panel en un solo bloque, para que sea fácilmente transportable, utilizando materiales sostenibles como la madera, en lugar de plásticos para el armazón.

El ordenador que controle el simulador, irá incorporado dentro de la propia estructura, al igual que el monitor que mostrará los indicadores del avión. Además, la propia estructura tendrá una forma que permitirá apoyar sobre ella, un segundo monitor más grande para mostrar las vistas exteriores del juego.

La estructura contará con departamentos en los que se podrá guardar el mando a distancia del monitor exterior y un teclado inalámbrico con TouchPad incorporado[8].

Para poder crear un panel llamativo, se utilizará el mismo sistema utilizado en el actual simulador de la EETAC: un panel de metacrilato transparente con un vinilo pegado por su parte trasera. Esta combinación de metacrilato y vinilo permite la reproducción de leyendas y colores necesaria para dar sensación de realismo y espectacularidad ya que el metacrilato es ligeramente brillante. Además, al pegar el vinilo en la parte trasera, será protegido por el metacrilato aumentando su durabilidad y evitando el desgaste.

La solución al presupuesto máximo, corresponde a una correcta elección de componentes y a la selección del sistema de tarjetas controladoras que harán funcionar el simulador, y determinarán, en gran medida, el precio final del mismo. En el mercado de los simuladores personales existen tres grandes sistemas de tarjetas controladoras, dos de ellas específicas para simulación y otra, muy utilizada también, que es más genérica y también se suele usar:

El **sistema IOCards** es un sistema específico para crear simuladores de vuelo. Tiene una gran cantidad de documentación y ejemplos. Se trata de tarjetas extremadamente versátiles, modulares y de precio reducido en comparación con sistemas como SimIO Boards.

El **sistema SimIO Boards**[9] también es un sistema creado específicamente para simulación aérea. Cuenta de un diseño modular más reducido que las IOCards y aunque pretenden integrar varias tarjetas en una sola, ocurre que

[8] https://www.logitech.com/es-es/product/wireless-touch-keyboard-k400-plus
[9] http://www.simioboard.com/

sea posible que no se necesiten funciones y se esté pagando por algo que no se vaya a utilizar. Los precios de cada tarjeta son muy superiores a las IOCards.

Arduino[10] es un sistema muy bien documentado, con una gran cantidad de ejemplos y una comunidad fiel bien formada. Si bien se usa en simulación aérea por su bajo precio, al tratarse de un sistema no pensado para ello, su programación y montaje a la hora de querer utilizarse para cabinas de simulación, lo hace muy difícil de implementar.

Por estos motivos, en el diseño tanto del simulador portátil, como en el que construí en la Universidad en 2013, **se escogieron las IOCards**. Dada su versatilidad y bajo coste son perfectas para un proyecto como este.

El simulador será completamente funcional y genérico, es decir, podrá simular cualquier avión monomotor y bimotor que se instale, en las sesiones de uso, siendo uno de los motivos por los cuales se ha optado por utilizar un monitor que muestre los instrumentos, en lugar de utilizar instrumentos reales de simulación (como en el caso del simulador de la escuela).

Para poder adaptar todas las características al presupuesto (muy bajo en comparación con cualquier simulador comercial de características similares), se ha optado por utilizar el sistema de IOCards y no poner indicadores reales, sino, el monitor que los simula. Además, se ha buscado los componentes necesarios en lugares en los que se ofrecían más económicamente. También, el carácter artesanal del simulador ayuda a no incrementar los precios finales.

A nivel de hardware, el simulador será robusto en las piezas de más desgaste, mediante unos buenos anclajes de los componentes y el diseño de la estructura con puntos de refuerzo.

A nivel de software se podrá garantizar un buen funcionamiento, gracias a mi experiencia por haber programado el simulador de la EETAC y mis otros dos simuladores personales, por haber trabajado durante años como técnico informático y por haber dado soporte a personas de todo el mundo sobre el tema de simulación aérea e IOCards. Además, el haber realizado mantenimiento del simulador de la escuela, desde 2013 hasta la actualidad, también me ha dado soltura para poder prevenir los posibles fallos y solucionar aquellos que se presenten de manera eficaz.

El simulador de vuelo portátil contará con una interfaz muy sencilla de utilizar, con vuelos pre-programados y todos los programas instalados necesarios para hacerlo funcionar.

Además, se intentará que el proceso de creación del panel y su elección de componentes respete el medio ambiente en la medida de lo posible, utilizando maderas en lugar de plásticos, y escogiendo componentes de bajo consumo, dentro de los límites de presupuesto.

[10] https://www.arduino.cc/

1.2 Diseño del panel de simulación en 2D

Para diseñar el panel, se ha utilizado el software vectorial en 2D, Adobe Illustrator CS 2017 en su versión de prueba gratuita.

El panel de simulación ha venido determinado en tamaño y disposición de sus componentes principalmente por el monitor interno que simula los distintos instrumentos de vuelo.

Es por ello que se ha tenido que delimitar mediante color rojo, los elementos que puedan interferir con los distintos componentes electrónicos que irán montados en el panel **(Fig. 1.1)**.

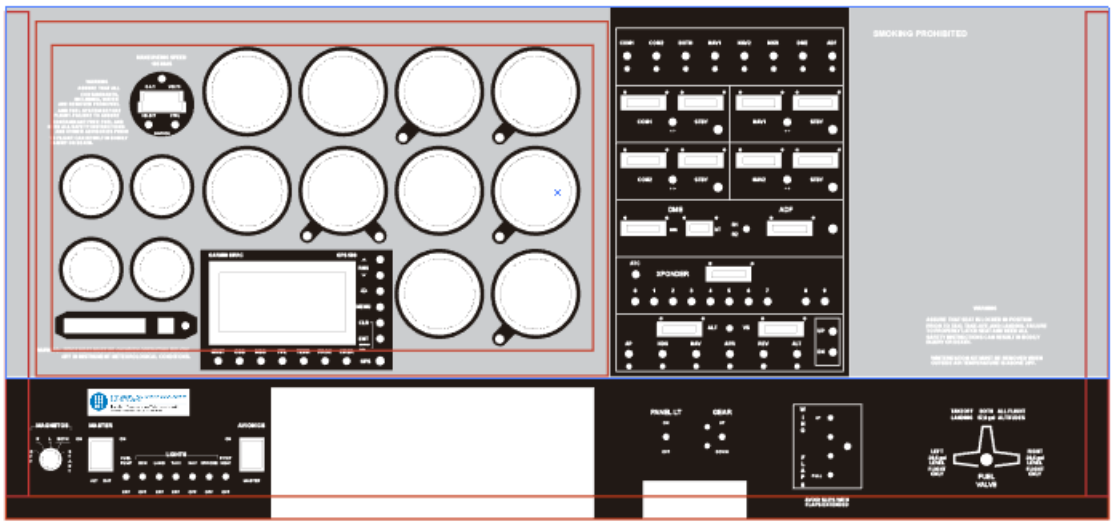

Figura 1.1: Vista general del panel con límites internos en rojo

Puede notarse que algunos componentes, principalmente encóders y los botones del GPS, se encuentran en el espacio del monitor, es por ello que se utilizarán componentes de perfil bajo y el monitor se separará unos milímetros del panel para que no rayen la pantalla o toquen con el marco.

El simulador se ha diseñado basado estéticamente en una cabina de Cessna 172 y para que cuente con las siguientes características:

- Instrumentación completa (gauges) de avionetas monomotor y bimotor simuladas en un monitor.
- Simulación de un GPS Garmin 500[11] completamente funcional.
- Panel de fallos.
- Cuadro de luces.

[11] https://buy.garmin.com/en-SG/digital/p/115

- Sección de control de Flaps, tren de aterrizaje y selector de tanque de combustible.
- Cuadro de radios de navegación y comunicaciones.
- Piloto automático completo y funcional.

Puede verse una vista general del panel con más detalle en las siguientes imágenes:

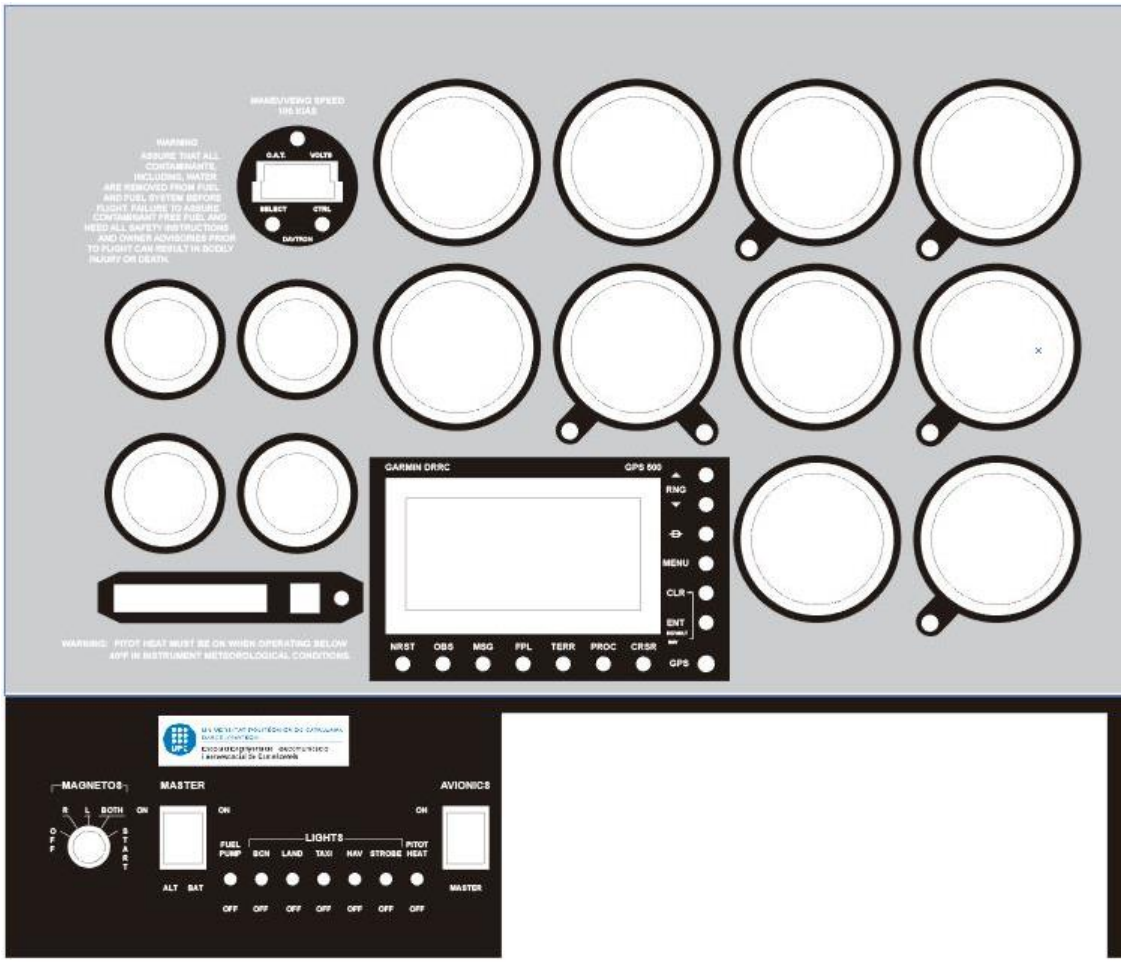

Figura 1.2: Vista de la sección izquierda del panel

En la imagen superior puede notarse que se ha dejado espacio en la parte inferior del panel, para dar cabida a un mando de control, en concreto el Saitek Pro Flight Yoke[12]. Puede verse el espacio superior, en el que va incorporado el monitor interno que simulará los instrumentos, y el GPS en un lugar céntrico y muy accesible.

[12] http://www.saitek.com/uk/prod-bak/yoke.html

En la parte inferior puede verse el logotipo de la escuela, para promocionar el centro mediante el simulador de vuelo y el panel del cuadro de luces y encendido del motor.

Aunque sólo se encuentre un interruptor para el motor, estará programado de manera que actúe en todos los motores a la vez en el caso de que se quiera volar con bimotores.

La siguiente imagen corresponde a la sección derecha del panel.

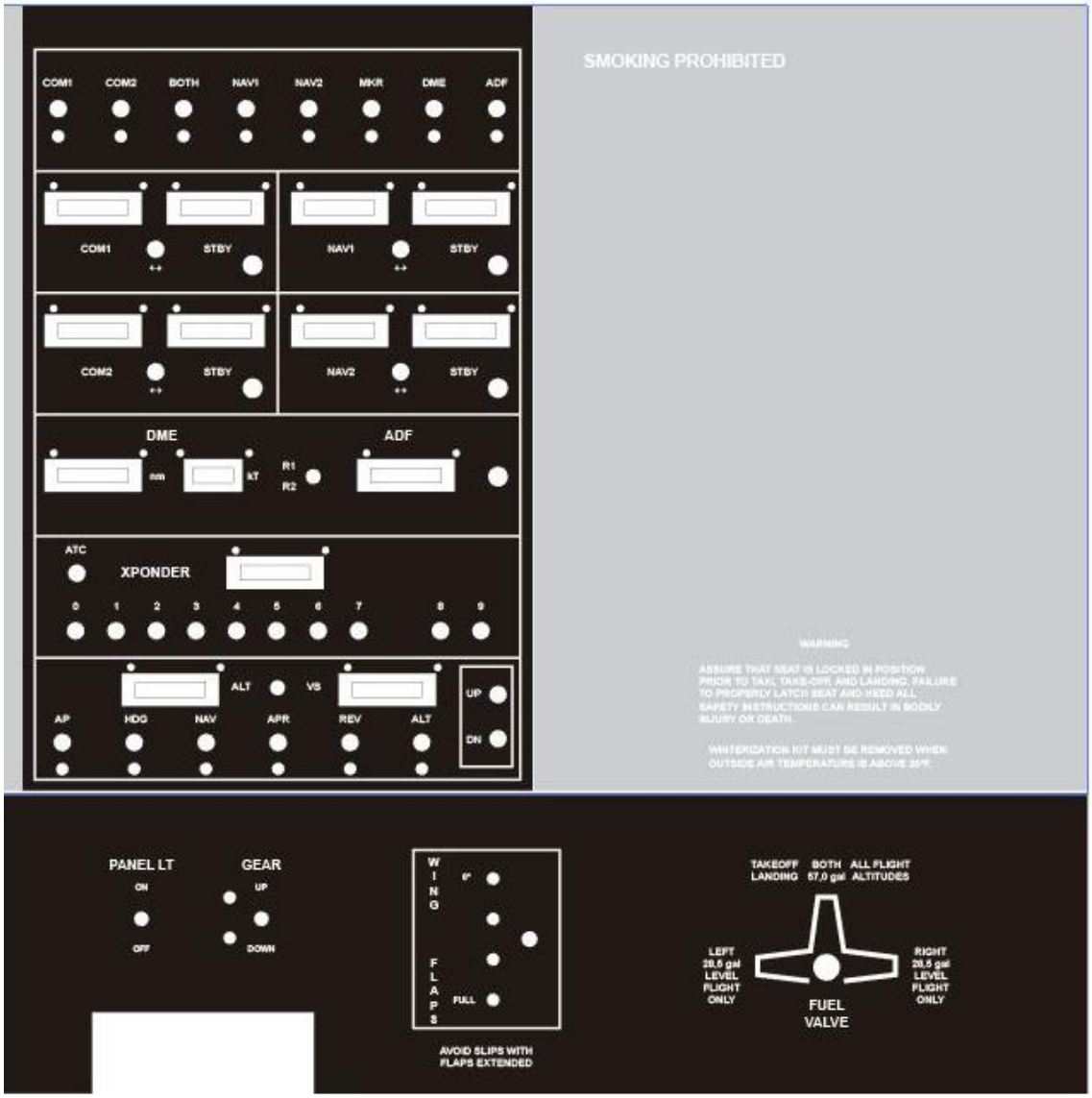

Figura 1.3: Vista de la sección derecha del panel

En esta sección se encuentran las radios y el piloto automático.

Las radios servirán para comunicarse con el ATC del propio simulador de vuelo, y para comunicarse con otros jugadores mediante internet por IVAO[13].
En la parte inferior puede verse el panel que controla los Flaps, el tren de aterrizaje y el selector de tanque de combustible.

En ambas imágenes puede notarse la cuidada disposición de todas las leyendas y la elección de colores y fuentes, para hacer el panel muy llamativo, realista y fácil de usar.

1.3 Diseño 3D del simulador portátil

Para realizar el diseño general del panel de simulación portátil, se ha pensado en la manera de que fuera más sencilla su portabilidad, por ello se ha decidido diseñar un panel de un metro de largo y 45 m. de alto con una estructura de madera con una forma tal que permita posicionar una pantalla sobre ella y en su interior, albergar el ordenador de control y el segundo monitor de instrumentos. Además, podrá posicionarse sobre cualquier mesa para hacer más sencillo su uso.

En la siguiente imagen **(Fig. 1.4)** puede verse una prueba de concepto de Sketchup[14] en el que se muestra la idea de diseño del simulador portátil una vez montado sobre una mesa y con el monitor de las vistas y los pedales instalados.

Figura 1.4: Vista delantera del concepto en 3D

[13] https://www.ivao.aero/
[14] https://www.sketchup.com/es

En la imagen siguiente, se puede ver mejor la forma del mueble que contendrá el ordenador y los monitores (el de instrumentos no se ve, va dentro del mueble).

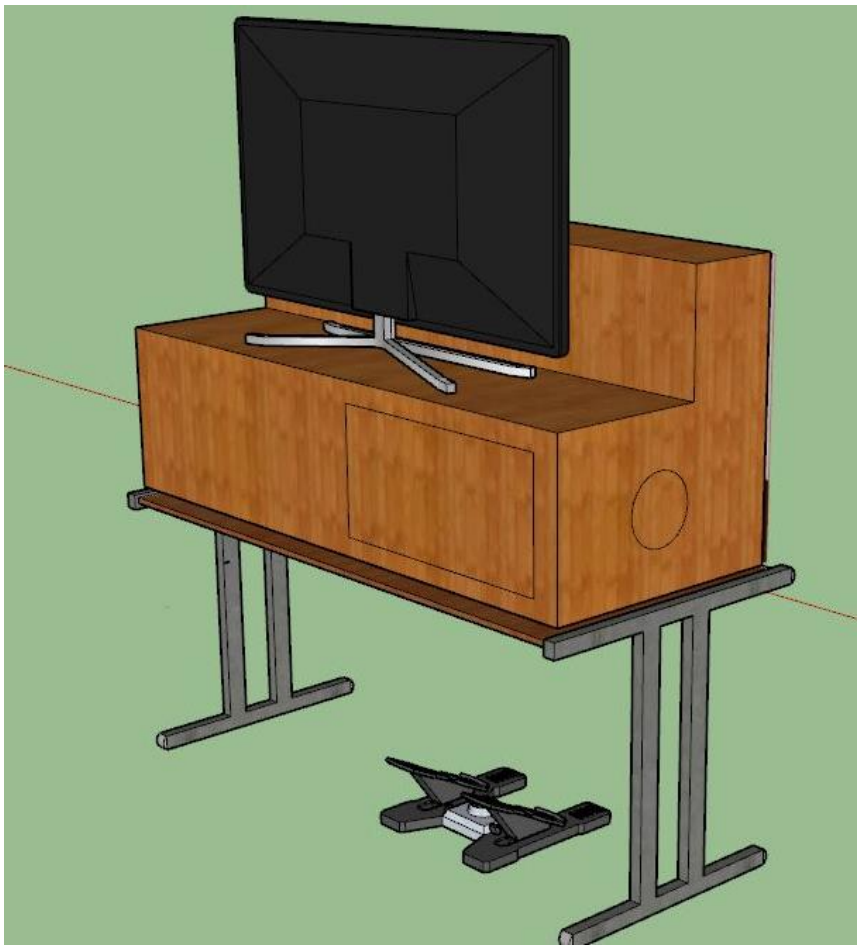

Figura 1.5: Vista trasera 3D de concepto

1.4 Funcionamiento general del sistema de simulación

El simulador portátil (incluyendo los monitores) irá gestionado a través de un ordenador integrado en su propia estructura.

El panel de simulación, con sus inputs y sus outputs digitales, podrá comunicarse con el ordenador mediante las tarjetas IOCards como puede verse en la siguiente imagen. **(Fig. 1.6)**

Como se verá en los capítulos de “Diseño de software”, el sistema IOCards se comunica con el odenador mediante un programa llamado FSUIPC y con el panel de simulación mediante SIOC.

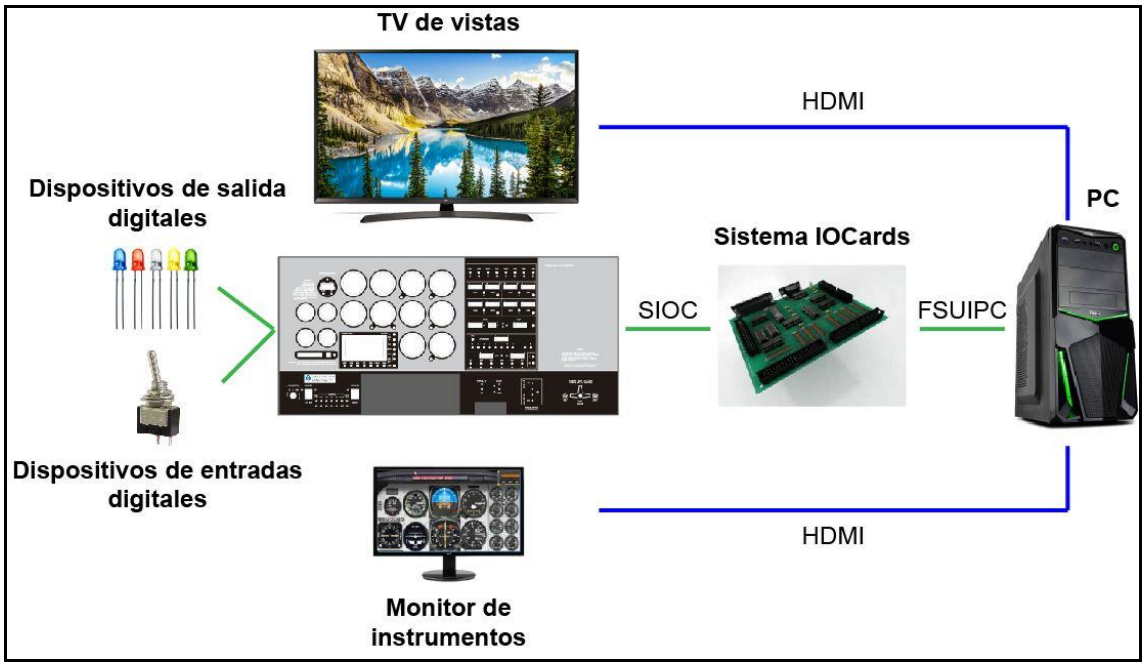

Figura 1.6: Esquema de componentes del simulador portátil

1.5 Sistema IOCards

Las IOCards son el alma de todo simulador de vuelo. Son las tarjetas electrónicas encargadas de hacer de interfaz entre el usuario (mediante interruptores, botones, potenciómetros, etc.) y las variables en memoria que controlan cada uno de los aspectos de un avión simulado.

Se trata de un conjunto de tarjetas electrónicas, cada una especializada en una o varias acciones relacionadas con la simulación. Existen tarjetas de todo tipo, como: las encargadas únicamente de gestionar entradas o salidas de datos (interruptores, leds, etc.), las que se encargan de mostrar dígitos en displays, otras capaces de accionar relés, servos, etc. Engloban prácticamente todas las acciones que se puedan necesitar a la hora de construir un simulador. Además, permite un diseño completamente modular y expandible, lo cual facilita el trabajo de diseño y posteriores mejoras.

En relación a la conexión con el PC, existen varias posibilidades. Dependiendo del tipo de tarjeta y cuál se escoja a la hora de comprarlas, pueden ir conectadas directamente al USB del ordenador, o pueden ir conectadas a una entrada de la tarjeta “Master” (la tarjeta principal).

Normalmente, las tarjetas que se conectan a la “Master” son las encargadas de gestionar las entradas (interruptores, botones, etc.) y las salidas de datos (displays o LEDs).

Las que se conectan directamente al PC por USB, suelen controlar aspectos más específicos, como la gestión de motores DC, AC, servos, o tarjetas

especiales. Además, estas últimas tienen un sistema de alimentación de corriente externo que aporta la potencia extra necesaria para cada acción.

La tarjeta “Master” proporciona una salida mediante un puerto paralelo que puede ser conectada a ordenadores (antiguos) que aún dispongan esa entrada. Aunque como veremos, no se recomienda hacerlo y es mejor conectarlas mediante USB.

Así pues, el conexionado de las IOCards (por puerto paralelo) sería el siguiente:

- **PC > Tarjetas USB**
- **PC > Master (por puerto paralelo) > Resto de tarjetas**

El puerto paralelo está cada vez en más desuso, por lo tanto, se desarrolló la tarjeta “USBExpansion”. Dicha tarjeta, transforma la salida paralela a la más actual y utilizada, USB, además de proporcionar conversores analógicos/digitales para poder incluir inputs analógicos al sistema, como por ejemplo: potenciómetros que controlen los mandos, pedales, rueda del trim, palancas de gases, etc.

Con lo cual, en el simulador de la EETAC y en el portátil, se sigue el siguiente esquema:

- **PC > Tarjetas USB**
- **PC > USBExpansion > Master > Resto de tarjetas**

Para hacer funcionar todas las tarjetas, se necesitará instalar y ejecutar en segundo plano el programa SIOC proporcionado por Opencockpits.

Se tendrá que realizar un proceso de configuración inicial mediante la modificación del archivo “Sioc.ini” en la carpeta de instalación de SIOC y vincular cualquier archivo programado en lenguaje SIOC.

1.5.1 Características de las tarjetas IOCards[15]

Cada tarjeta IOCard está especializada en algún aspecto concreto utilizado en simulación aérea, a continuación, se describirán las tarjetas utilizadas en el diseño propuesto.

[15] http://www.opencockpits.com/index.php/es/descargas/category/iocards

1.5.1.1 Tarjeta USBExpansion

La tarjeta USBExpansion **(Fig. 1.7)** es una tarjeta de carácter opcional pero altamente recomendada.

Permite conectar el resto de tarjetas directamente al puerto USB, en lugar del puerto paralelo y mejorar la configuración y expansión del grupo de tarjetas IOCards.

Permite la conexión de 4 masters y hasta 4 potenciómetros, además, pueden conectarse tantas USBExpansion como se quiera a un mismo PC incrementando la conectividad y las características prácticamente cuanto se necesite.

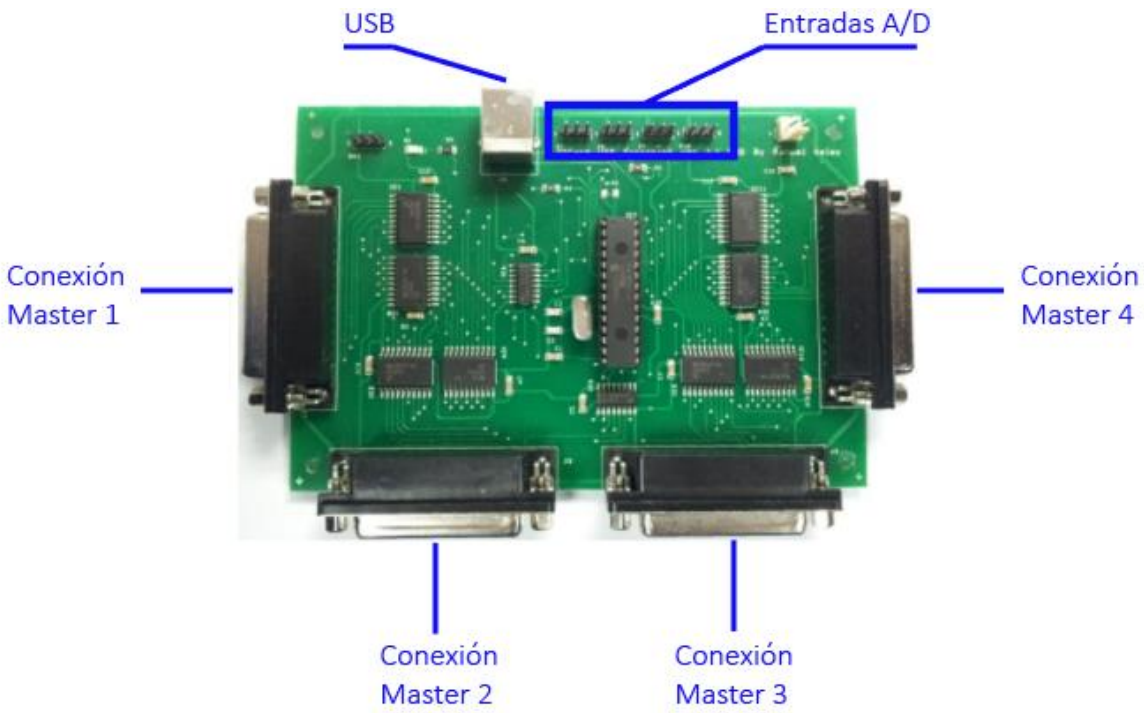

Figura 1.7: Tarjeta USBExpansion y sus conexiones

Características

- Se conecta por USB al PC
- En un mismo PC pueden conectarse tantas USBExpansion como se quiera.
- Pueden conectarse a ella hasta 4 tarjetas master
- No necesita alimentación
- Tiene 4 conversores A/D (entradas analógicas) para potenciómetros de 10k Ω

1.5.1.2 Tarjeta Master

La tarjeta Master **(Fig. 1.8)** es el cerebro de las tarjetas IOCards. Es la encargada de gestionar el resto de tarjetas conectadas a ella. Además, proporciona entradas y salidas digitales que pueden ir conectadas directamente a ella sin necesidad de otras tarjetas, lo cual permite más personalización a la hora de gestionar los cables; o bien, mediante el uso de las tarjetas de entradas y de salidas, con lo que se facilita el conexionado a costa de añadir nuevas tarjetas.

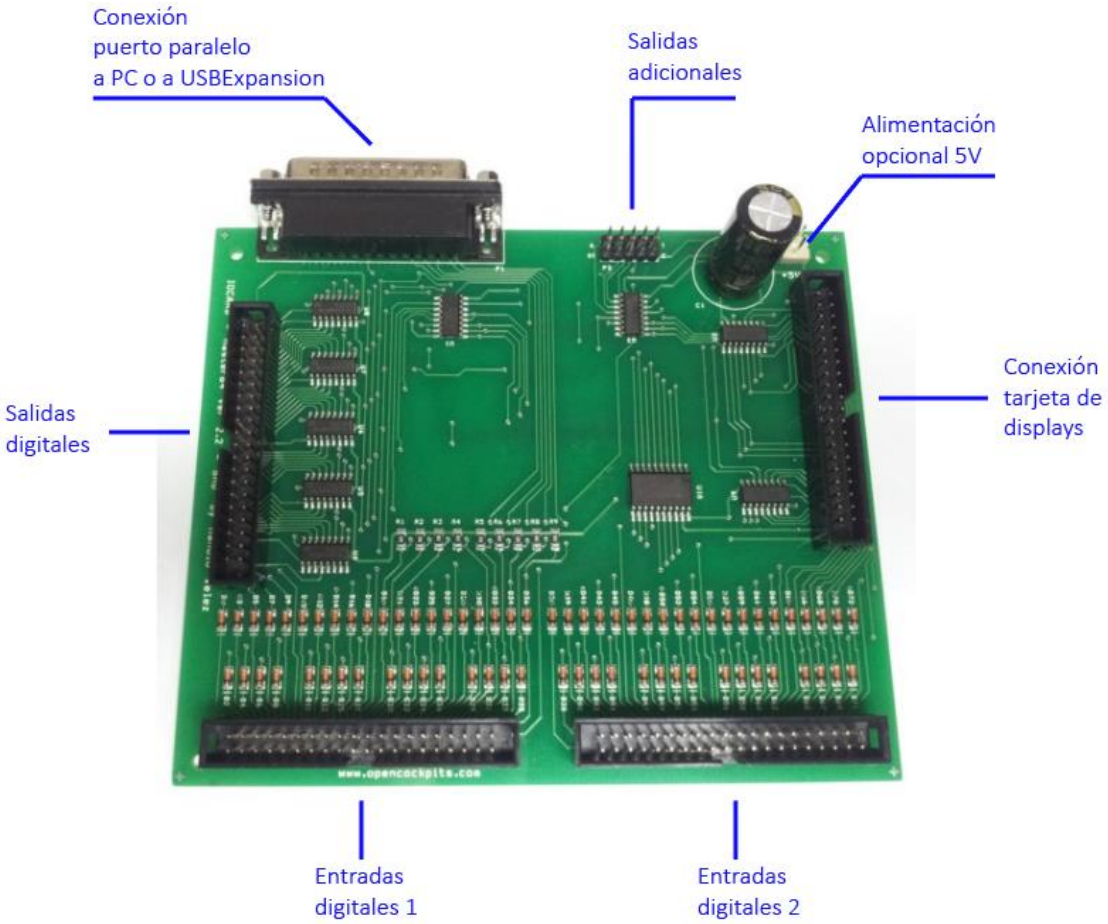

Figura 1.8: Tarjeta master y sus conexiones

Características

- Puede conectarse a la USBExpansion o al PC mediante el puerto paralelo
- Dispone de hasta 45 salidas digitales (38+7), donde conectar LEDs.
- Tiene 72 entradas digitales (para conectar interruptores, encóders, etc.)
- Permite su conexión a otras tarjetas IOCards, normalmente perdiendo entradas.

- Dispone de un conector específico que puede conectar hasta 4 tarjetas de Displays sin perder ninguna entrada
- No necesita alimentación. En el caso de usarse tarjetas de displays, sí que deberá alimentarse mediante una línea externa de 5V.
- Tiene la función de emulación del teclado en todas sus entradas.

1.5.1.3 Tarjetas I/O (Entradas o Salidas)

Las tarjetas I/O **(Fig. 1.9)** son tarjetas opcionales, pero altamente recomendadas, que facilitan la conexión tanto de las entradas como las salidas, es decir, de los botones, encóders, interruptores, etc. y los LEDs.

Aunque la tarjeta de Entradas y la tarjeta de Salidas son prácticamente iguales, no es así en su conexionado interno, así que hay que tener precaución de utilizar la tarjeta adecuada en cada momento.

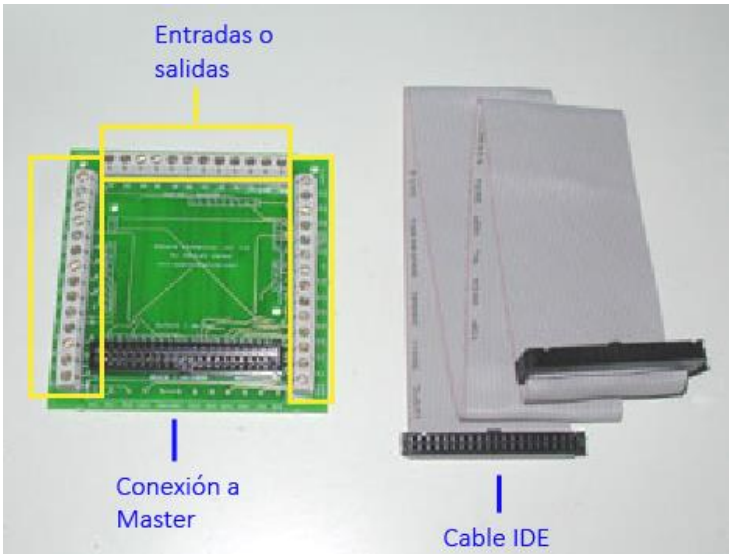

Figura 1.9: Tarjetas de salidas y entradas y cable utilizado

En la imagen, podemos ver todo un array de conexiones de color blanco, que corresponde a las entradas o salidas hacia nuestro hardware del simulador en función del tipo de tarjeta que se haya comprado.

Características

- Se conectan a la placa Master mediante el cable IDE de la **Fig. 1.9**
- Permite conexiones de los cables mediante tornillos
- Proporcionan entradas o salidas dependiendo de qué tarjeta se trate

1.5.1.3.1 Tarjeta de entradas

La tarjeta de entradas **(Fig. 1.10)** permite la conexión directa mediante tornillos, de hardware de entrada digital, como por ejemplo botones, encóders de 2 bits, interruptores, etc.

No permite la introducción de valores analógicos, por lo tanto, en esta tarjeta no es posible conectar potenciómetros.

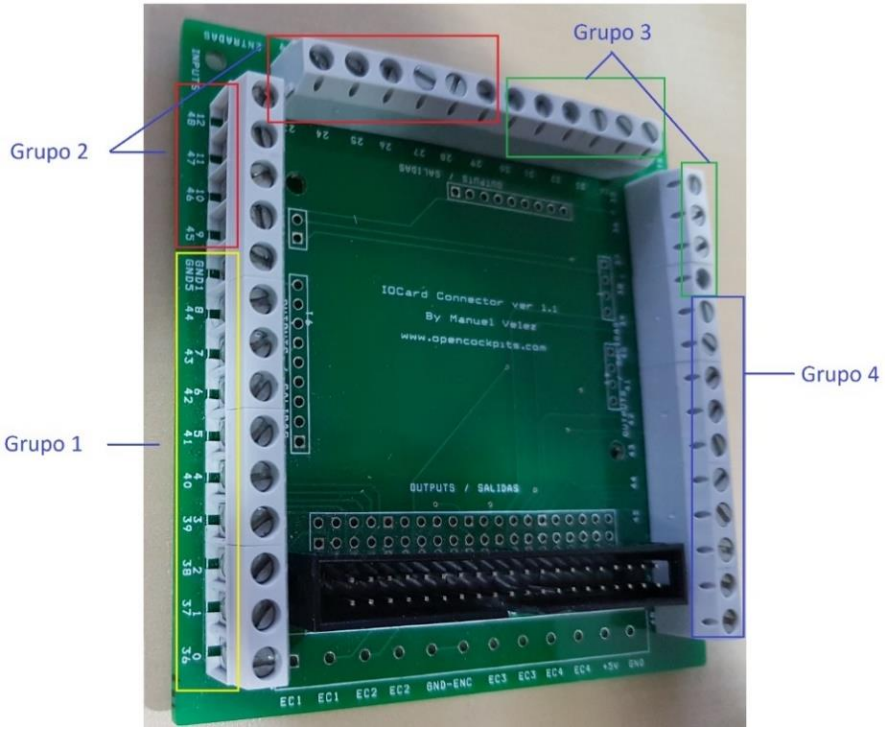

Figura 1.10: Tarjeta de entradas y sus conexiones

Como puede verse en la **fig.1.10**, estas tarjetas, dividen las entradas en grupos de 9 en 9. Cada grupo comparte su propio GND. Es decir, cualquier conjunto de interruptores puede soldarse de manera que una de sus dos patillas de conexión comparta la misma masa.

El resto de patillas irán soldadas a cada una de las salidas de la tarjeta, numeradas para su posterior programación.

Características:

- Hasta dos tarjetas de entradas por Master
- Cada una proporciona 36 (0 a 35 y 36 a 71) entradas digitales (72 en total)
- Dividida en subgrupos con masas en común (explicado a continuación)

1.5.1.3.2 Tarjeta de salidas

La tarjeta de salidas **(Fig. 1.11)**, permite la conexión directa de LEDs o la exportación de valores digitales a otras entradas, mediante tornillos.

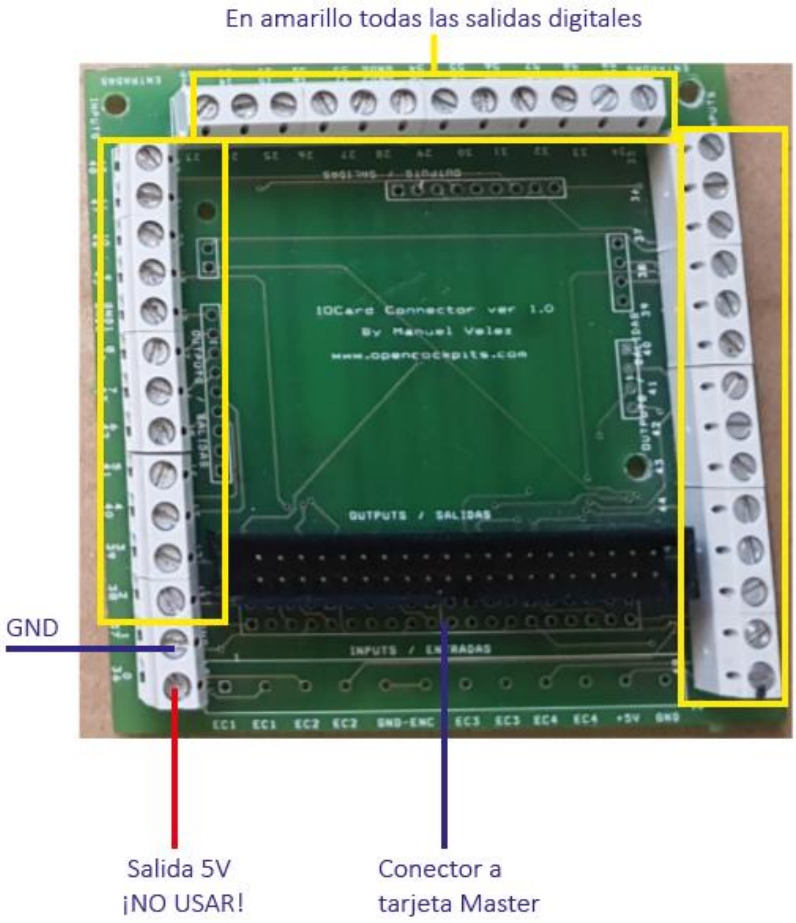

Figura 1.11: Tarjeta de salidas y conexionado

Características:

- Proporciona 38 salidas digitales numeradas del 11 al 48
- Tiene un solo GND o masa común

Como puede verse en la imagen, dispone de un único pin de masa común, por lo que todos los dispositivos de salida de datos digitales, podrán conectar sus masas a ese pin, independientemente del número.

Nunca deberá utilizarse la salida de 5V para no dañar la tarjeta (recomendado por el fabricante).

1.5.1.4 Tarjeta Displays

La tarjeta de displays **(Fig. 1.12)** es la encargada de mostrar números en displays de 7 segmentos (y el punto decimal si fuera necesario).

Se utiliza principalmente en la sección de las radios del avión.

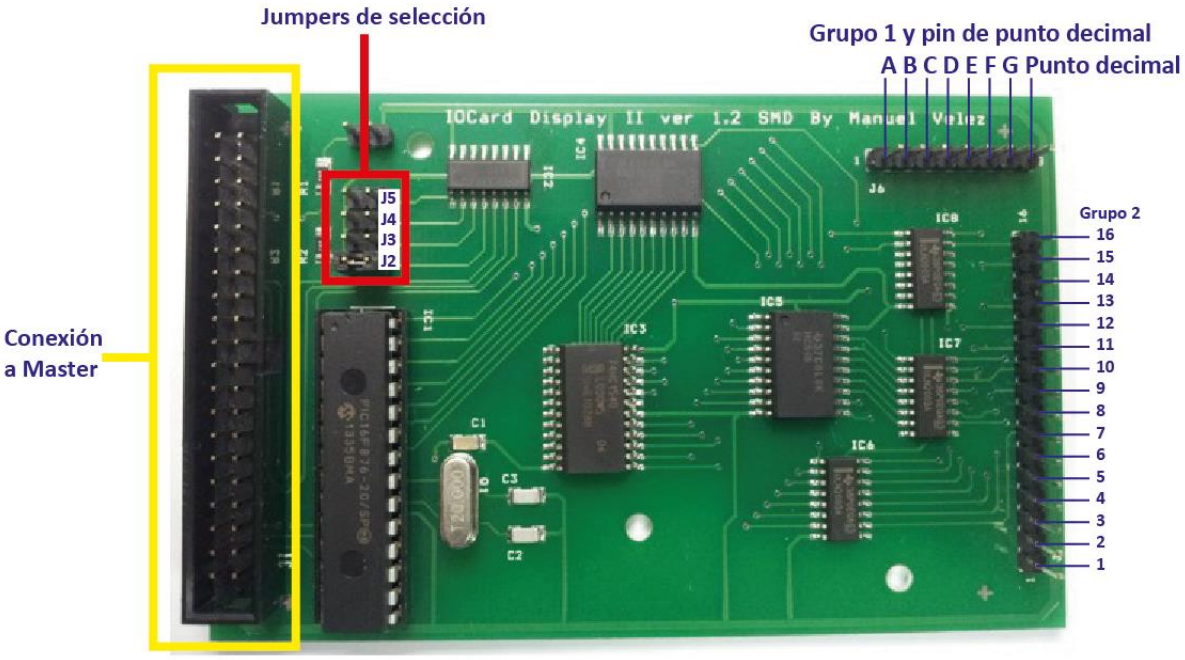

Figura 1.12: Tarjeta de Displays y sus conexiones

Características:

- Controla hasta 16 displays de 7 segmentos
- Se pueden conectar 4 a una misma tarjeta Master (64 displays por Master, 256 por USBExpansion)

Selección de número de tarjetas Display

A cada tarjeta Master, pueden ir conectadas hasta 4 tarjetas Display, por lo tanto, para poder identificar correctamente cada Display, se deberá, se deberá asignar un número distinto a cada tarjeta Displays utilizada, para que funcionen correctamente.

La selección se hará mediante un “jumper” que tiene la misma tarjeta **(Tabla 1.1)**.

Si sólo utilizamos una tarjeta de Displays, se deberá seleccionar la posición J2 correspondiente a la primera tarjeta. Si se usan, por ejemplo, 3 tarjetas, una de

ellas tendrá que ser J2, la otra J3 y, por último, J4 en la restante. A continuación, se puede ver una tabla mostrando las identificaciones para cada tarjeta y su jumper correspondiente.

Tabla 1.1. Selección de ID de tarjeta mediante Jumper

Jumper seleccionado	N.º de tarjeta	ID displays
J2	1	0-15
J3	2	16-31
J4	3	32-47
J5	4	48-63

1.5.2 Elección de tarjetas

El simulador portátil, se ha previsto que necesitará utilizar las siguientes tarjetas IOCards:

- Entre 3 y 4 tarjetas de entradas para proporcionar hasta 144 entradas digitales, es decir la conexión de hasta 144 botones, interruptores, etc. (el número exacto puede variar en función de si se necesita programar algún extra no previsto).
- 1 tarjeta de salidas, que proporcionará hasta 38 salidas digitales, perfectas para mostrar elementos luminosos mediante LEDs.
- 2 tarjetas Master.
- 1 tarjeta “USBExpansion”.
- 4 tarjetas de Displays: para mostrar todos los displays de 7 segmentos que formarán las radios del avión.

A continuación. puede verse el esquema del sistema IOCard que utilizará el simulador portátil, y que suele ser un esquema tipo para cualquier montaje de un simulador de avioneta completo. Para montajes más complejos, como por ejemplo simuladores de aviones comerciales, aparecerían más número de tarjetas, aunque todas seguirían el mismo esquema.

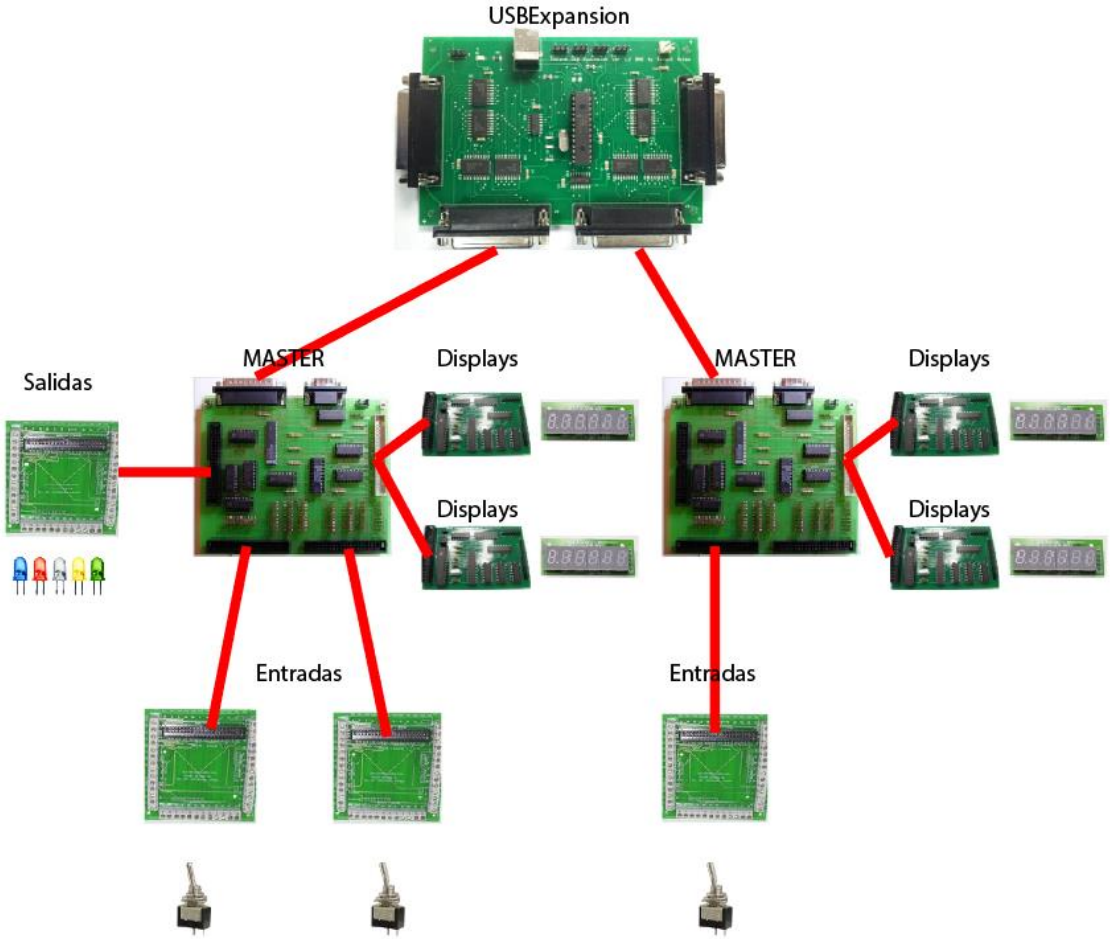

Figura 1.13: Sistema de tarjetas IOCards en el simulador portátil

Puede notarse que siguen un esquema tipo árbol, en el que las tarjetas controladoras principales (Master) se conectan mediante la USBExpansion, que únicamente actúa como concentradora y gestora de prioridades y se conecta al PC mediante USB.

1.5.3 Conexión general

Al no existir todavía un panel físico, la siguiente información sobre conexión, será genérica y la manera correcta de conectar los distintos dispositivos a sus respectivas tarjetas IOCards.

A continuación, se muestran varios ejemplos de conexión los componentes más utilizados, en la tarjeta de entradas:

1.5.3.1 Interruptores, pulsadores o botones

Para poder conectar elementos de entrada de datos, como, por ejemplo: se deberán conectarse todas las masas (GND) juntas y se atornillarán a la entrada GND correspondiente a cada grupo de entradas. El cableado de cada positivo de cada interruptor, deberá atornillarse dentro de su grupo de GND correspondiente, uno a cada entrada.

Se deberá hacer siguiendo el esquema de la **fig. 1.14.**

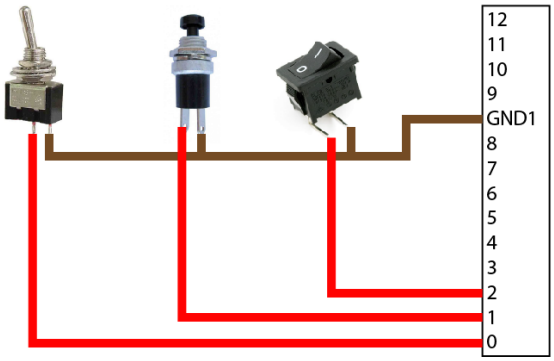

Figura 1.14: Ejemplo de conexión correcta de interruptores y botones

A la hora de programar en SIOC, se deberá recordar únicamente el número de la entrada a la que van atornillados.

1.5.3.2 Interruptores rotativos

Como puede verse en la **(Fig. 1.15)**, conectar un switch rotativo no difiere de cualquier otro interruptor.

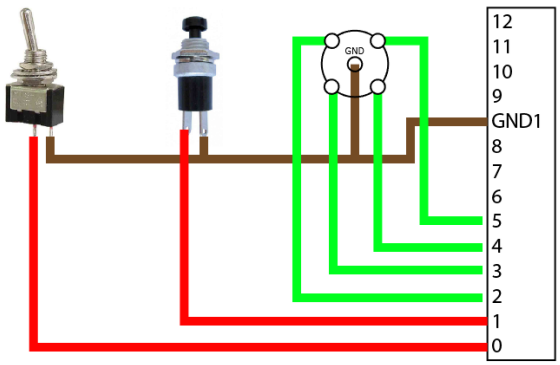

Figura 1.15: Conexión correcta de interruptores rotativos

Su masa tiene que ir común a la de su grupo y cada posición del interruptor rotativo va a una entrada del mismo grupo que la masa que se ha conectado.

Igual que antes, no se podrá conectar entradas en distintos grupos de GND si comparten masa.

Se ha añadido los dos interruptores al esquema para que se pueda ver que no influye qué conectemos a cada grupo mientras se respeten las masas.

1.5.3.3 Encóders

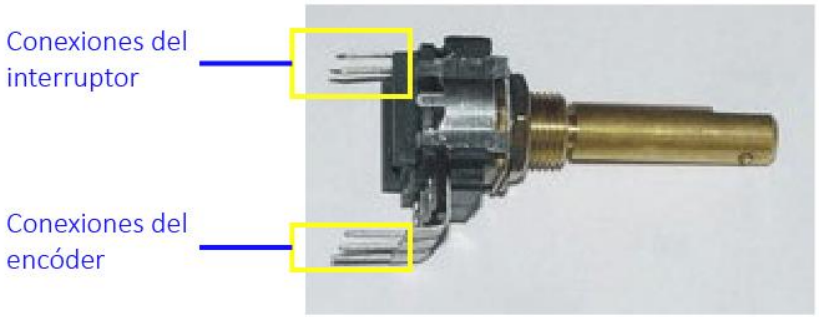

Figura 1.16: Encóder con pulsador

Aunque existen tarjetas especializadas en encóders, éstos pueden conectarse directamente a la tarjeta de entradas, aquellos encóders con las siguientes características:

- Tipo Gray
- 2 bits
- ¼ de ciclo por detención

Además, puede tratarse de encóder + interruptor **(Fig. 1.17)**, como los utilizados en el simulador de la EETAC y en el simulador portátil.

Cada encóder gasta dos entradas y una masa (GND). En caso de incorporar interruptor, además, gastará una entrada y masa adicionales. Las dos masas, en ese caso pueden ir soldadas juntas.

Como puede apreciarse, la parte del encóder tiene 3 pines, siendo el del medio la masa. Los dos superiores son del interruptor incorporado, siendo cualquiera la masa y el que queda, el de señal.

La conexión será la siguiente:

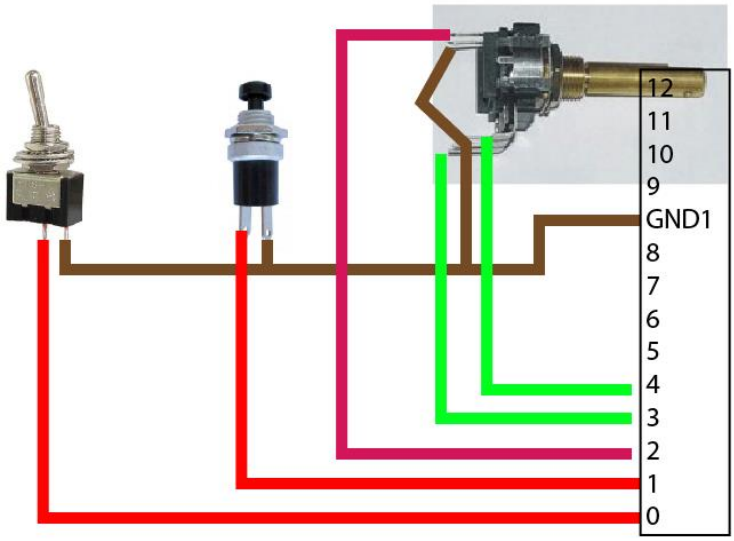

Figura 1.17: Conexión correcta de un encóder con pulsador

De nuevo, se han añadido los dos interruptores al esquema para que se vea que no influye qué conectemos a cada grupo mientras se respeten las masas.

En el caso de un encóder sin interruptor, sólo se tendrán los tres pines (verdes y marrón) propios del encóder en vez de los cinco de la imagen.

Displays de 7 segmentos

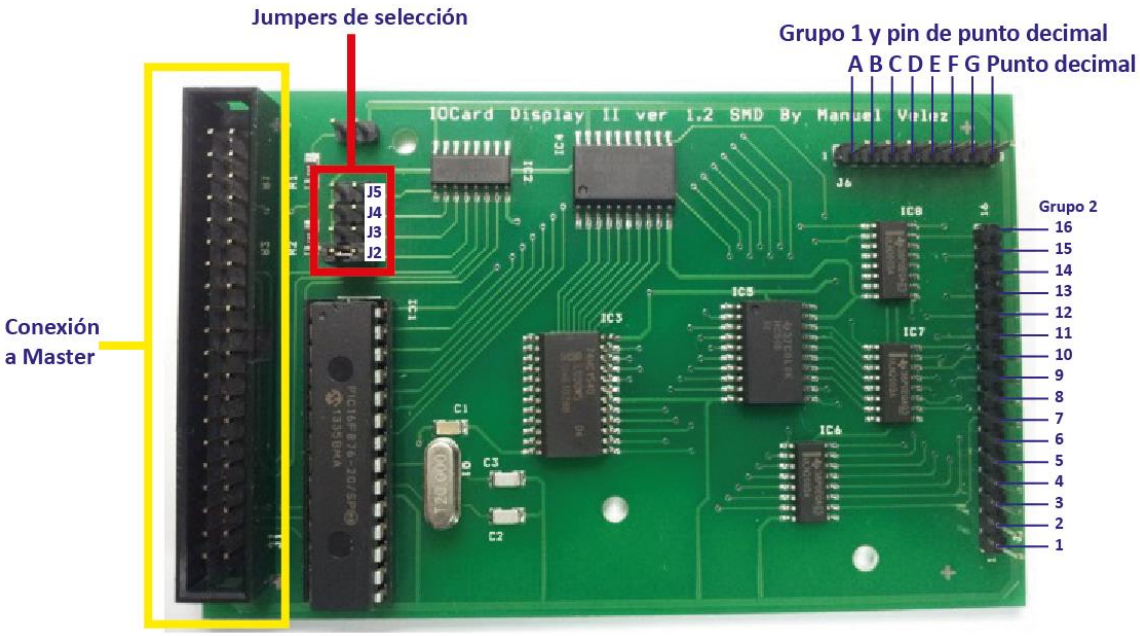

Figura 1.18: Número de pines de la tarjeta "Displays"

La tarjeta Displays **(Fig. 1.18)** dispone de dos grupos de conectores: el primero de la “A” a la “G” y el segundo, del “1” al “16”. Nótese que llamamos al segundo grupo “1” a “16”, pero dependiendo de la posición del “jumper”, corresponderá a las entradas 0 a 15, 16 a 31, etc.

De la misma manera, la placa PCB de displays (**Fig. 1.19**), también tiene dos grupos de conectores: en este caso se diferenciarán mediante el uso de minúsculas. De la “a” a la “g” y del “D1” al “D3” (PCB de 3 displays), del “D1” al “D5” (PCB de 5 displays) y “D1” al “D6” (PCB de 6 displays).

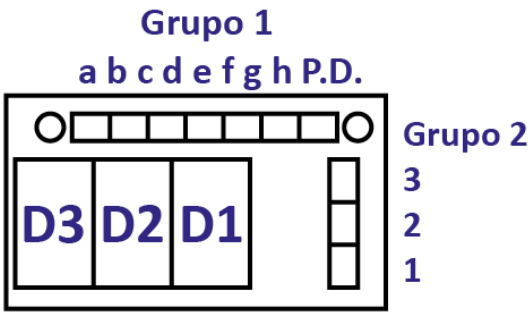

Fig. 1.19: Diagrama de conexión de PCB de Displays

El primer grupo (A-G) irá conectado al grupo vertical (a-g) de la PCB de los displays. Se trata de una conexión común, es decir, de un solo grupo (A-G) de la tarjeta Displays, irá al resto de grupos (a-g) de los PCB de displays.
Nunca se deberán conectar los primeros grupos de las TARJETAS entre sí, cada tarjeta puede gestionar hasta 16 displays, no más.

El segundo grupo corresponderá un pin para cada Display. Es decir, en este caso cada Display dispondrá de únicamente una posición en el segundo grupo de la tarjeta.

Teniendo en cuenta los puntos anteriores, un ejemplo de conexión correcta de dos PCB de displays (de 6 y de 3 dígitos) será el siguiente **(Fig. 1.20)**:

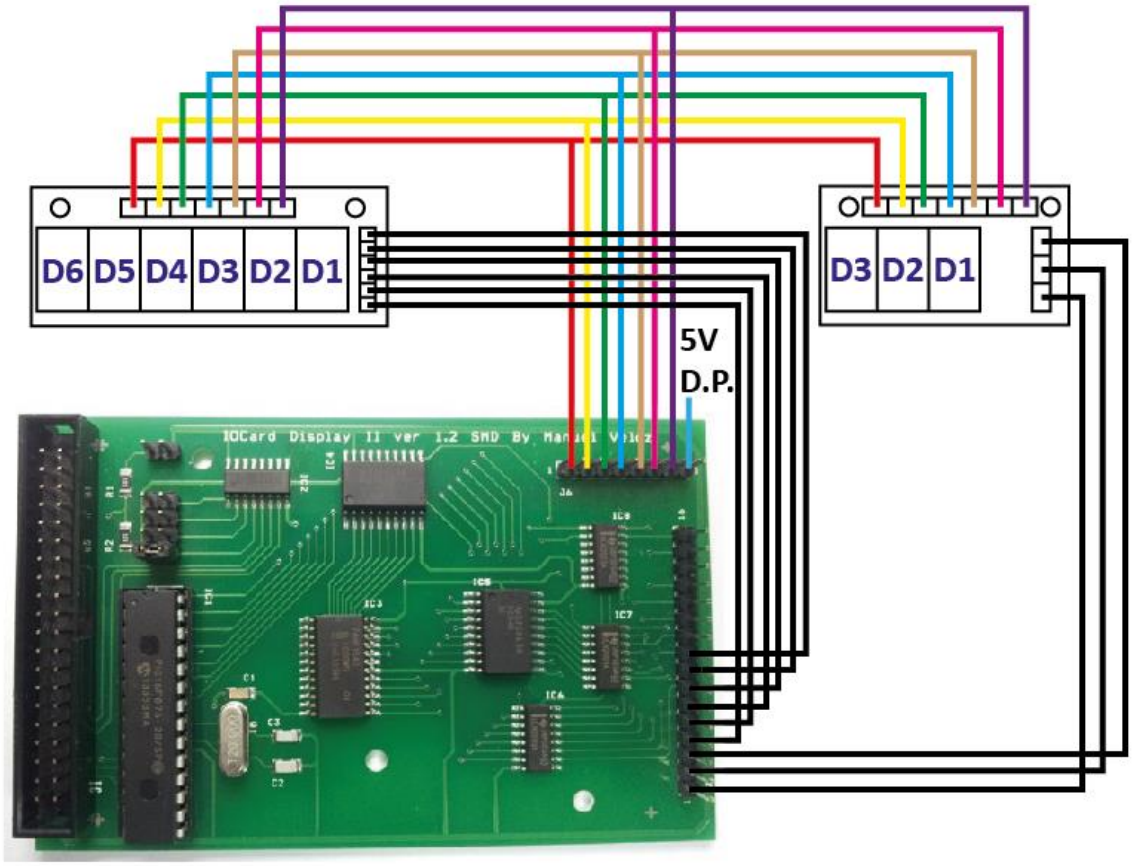

Figura 1.20: Ejemplo de conexiones de 2 PCB de Displays diferentes

1.6 Presupuesto

Teniendo en mente todos los objetivos y características demandadas y descritas anteriormente, se ha escogido los siguientes componentes:

Tabla 1.2. Listado de componentes de ordenador

	Cantidad	Nombre	Precio
Placa Base	1	Asus B150M Pro Gaming	82,99 €
Procesador	1	I5 7600K 3,8GHz	205,00 €
Disco SSD	1	Sandisk SSD Plus 120 GB Sata III	63,95 €
Gráfica	1	Gigabyte GTX 1060 WINDFORCE OC 3GB GDDR5	225,00 €
Memoria	1	G.Skill Ripjaws V Red DDR4 2666 PC4-21300 8GB 2x4GB CL15	111,00 €
Caja	1	Owlotech EVO USB 3,0 500W	39,95 €
Ventiladores adicionales	2	Enermax T.B. Silence PWM 120x120x25	25,90 €
Ventilador CPU	1	Nox Hummer H-120 CPU Cooler	13,95 €
Monitor	1	Viewsonic VA2261H-8 22" Led FullHD	89,99 €
Televisor	1	TV Philips 42PFS4012 42" LED FullHD	279,00 €
Conector adaptador	1	Unotec Adaptador DisplayPort a HDMI	8,75 €
Sistema operativo	1	Windows 10 Home 64 Bits OEM	105,00 €
Periférico: teclado + touchpad	1	Logitech K400+ Wireless Touch Keyboard Negro	32,95 €
		Total sin portes	1.283,43 €

Tabla 1.3. Listado de periféricos

Mandos	1	Saitek Logitech Pro Flight Yoke System	137,00 €
Pedales (opcionales)	1	Saitek Logitech Pro Flight Rudder Pedals	129,00 €
		Total sin portes	266,00 €

Tabla 1.4. Programa simulador de vuelo

Programa simulador	1	Prepar3D V4 **Academic License**	59,95 $
		Total sin portes	59,95 $

Tabla 1.5. Componentes IOCards

Nombre	Cantidad	Precio
Encoder con pulsador	12	53.88 €
Pulsador redondo negro	60	60,00 €
Led Ámbar 5 mm (5 uds)	5	5,00 €
Led Verde 5 mm (5 uds)	1	1,00 €
Interruptor MON-OFF-MON	1	2,00 €
Digitos 7 seg. Montados (3 Digits.) Color: Ambar	2	18,00 €
Digitos 7 seg. Montados (5 Digits.) Color: Ambar	15	180,00 €
Interruptor rotativo 12 posiciones	1	3,00 €
Botón gris estándar	12	28.80 €
Conexión entradas	4	72,00 €
Tarjeta USB Expansión	1	38,00 €
Conexión salidas	1	18,00 €
Tarjeta Displays II	4	132,00 €
Cable para tarjeta Displays	2	6,00 €
Tarjeta Master	2	110,00 €
Cable paralelo	2	14,00 €
	Total sin portes	741,68 €

- **Metacrilato y vinilo**

A determinar cuando la escuela permita su presupuesto. Se prevé un precio menor a 300 €.

- **Presupuesto aproximado**

Al tratarse de tiendas online, las cuales hay que sumar cargos por portes y en el caso de Opencockpits no incluyen IVA, y a falta de determinar el presupuesto final del metacrilato y el vinilo, el presupuesto aproximado del simulador portátil rondará los 2800 € y no se prevé superar los 3000 €.

Se trata de un precio muy por debajo del simulador de vuelo más económico y con menos funciones que el diseñado para la escuela: el simulador Prop Cockpit Trainer de VRInsight a la venta por 4900 €[16]).

[16] https://www.wilcopub.com/prop-cockpit-trainer.html

Capítulo 2. Diseño de software

Tan importante como el hardware de un simulador, es la parte que lo controla: el software. En el caso del simulador portátil constará de tres partes o programas diferentes:

- La primera es el programa de simulación propiamente. En el caso del simulador de la escuela es **Flight Simulator X** y en el caso del panel portátil será **Prepar3D**[17], aunque será cien por cien compatible con Flight Simulator X.
- La segunda parte es la que se encarga hacer funcionar el hardware y realizar los algoritmos que hayamos programado para cada palanca o interruptor. En concreto hablamos de los **drivers SIOC**[18] mediante el lenguaje de programación con el mismo nombre, que llamaremos "lenguaje SIOC" para diferenciar.
- Por último, se necesitará un programa que haga de intermediario entre los drivers y FSX. Este programa intermediario será **FSUIPC**[19]. Y es el encargado de leer y escribir las variables en memoria.

Para hacer funcionar el simulador de vuelo se deberá disponer pues, de un sistema Windows (cualquier versión y arquitectura) y los siguientes programas instalados en este orden:

1. **Programa simulador de vuelo (Prepar3D)**
2. **FSUIPC**
3. **Programario SIOC**

Una vez instalados deberá realizarse la configuración del archivo SIOC.ini que se encuentra en la ruta de instalación de SIOC tal y como se verá en el siguiente apartado.

Drivers SIOC

El programa SIOC.exe (**Fig. 2.1**) recibe y modifica las variables proporcionadas por FSUIPC mediante el protocolo IOCP, dependiendo de las condiciones que se hayan definido mediante su lenguaje de programación, en lenguaje SIOC. Es decir, es el encargado de leer el estado de todo el hardware del simulador y actuar sobre las variables de FSX mediante FSUIPC.

[17] https://www.prepar3d.com/
[18] http://www.opencockpits.com/index.php/en/download/item/sioc-ver-50b5?category_id=39
[19] http://www.schiratti.com/dowson.html

También realiza el sentido contrario: mostrar dígitos en displays, actuar sobre servomotores, etc.

Además, sirve de driver de las IOCards y es el encargado de que funcionen bajo Windows.

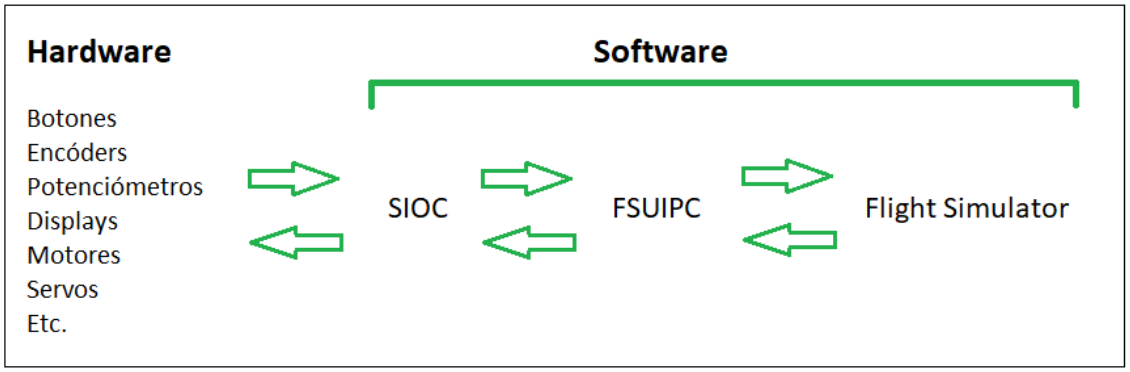

Figura 2.1: Interfaz SIOC

FSUIPC

Desarrollado por Peter Dowson, se trata de un programa que lee y modifica todas las variables de memoria que tengan que ver con cualquier proceso de los aviones instalados por defecto en FSX.

Además, el Sr. Dowson publica regularmente unas tablas en PDF, que se agregan con la instalación del programa, en su carpeta de instalación por defecto, en los que aparece un listado de todas las variables conocidas y las que va descubriendo mediante ingeniería inversa.

Estas tablas se encuentran en el directorio de instalación por defecto del programa FSUIPC en los siguientes temas se verá cómo encontrar las variables necesarias para la programación del simulador portátil.

2.1 Programación del simulador portátil

Para realizar la programación de la lógica que controla todas las tarjetas IOCards, se ha utilizado el sistema de programación que incorpora el programario de SIOC, en concreto el lenguaje de programación SIOC.

Al igual que las tarjetas IOCards, ha sido desarrollado por Manuel Vélez y se puede encontrar toda la información de uso en su página Opencockpits.

Al no poder disponer del panel físicamente, las entradas se han programado con un valor al azar y una vez construido se tendrán que adecuar a la identificación real.

Podrá verse todo el código programado en el **(Anexo 1. Código)**

También es posible que una vez montado, alguna línea de código tenga que ser modificada.

Dicha programación permite el control de los siguientes procesos:

Panel de instrumentos

- Botones reloj Davtron: selector volts, select y ctrl
- Ventana de Kollsman del altímetro
- Selector de radiales NAV1 y NAV2
- Selector de radial de la brújula magnética
- Selector HDG
- Selector de radial del NDB
- Selector NAV/GPS
- Botones de GPS
- Encóder y botón integrado del GPS

Cuadro de luces, arranque del motor y controles de vuelo

- Interruptor rotativo para uso como arranque de motor
- Botones de alternador, baterías y aviónica.
- Bomba de combustible
- Conmutadores de luces
- Calentador del tubo de Pitot
- Conmutador de luz de instrumentos
- Interruptor para tren de aterrizaje
- Pulsador de Flaps (e indicadores)
- Interruptor rotativo para el selector de tanque de combustible

Grupo de radios y piloto automático

- Panel de audio (e indicadores)
- Panel COM/NAV 1
- Panel COM/NAV 2
- Panel DME y ADF
- Panel de transpondedor (xponder)
- Panel de piloto automático con selector de altitud y velocidad vertical

Para poder realizar la programación en lenguaje SIOC, se ha debido estudiar antes su funcionamiento y manera de proceder. Es por ello que se dedicará este tema y sus siguientes sub apartados al estudio del lenguaje SIOC que se ha necesitado aprender para el diseño del simulador portátil. Además, todos los procesos se programan de la misma manera que se explicará en el siguiente apartado.

La programación en lenguaje SIOC se realiza mediante el editor (recuadro amarillo que incorpora el programa "SIOC.exe" (**Fig. 2.2**).

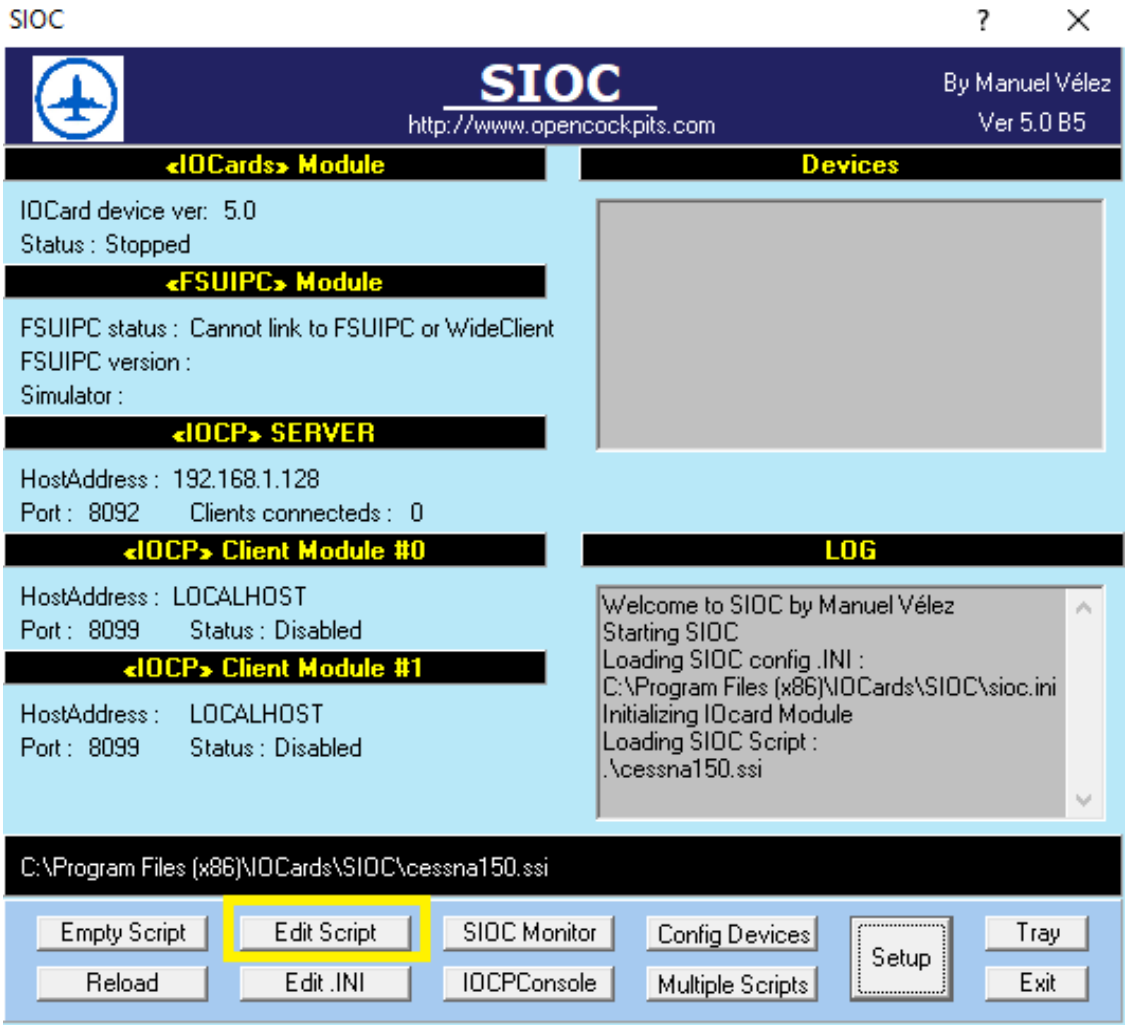

Figura 2.2: Programa SIOC.exe y acceso a edición de programación

Se trata de un método de programación mediante interfaz gráfica y "Point and Click", es decir, no se necesita escribir líneas de código, como en la programación tradicional, sino que permite el ingreso a través del ratón de los comandos necesarios mediante una interfaz gráfica.

La programación en SIOC consiste en líneas de código que controlan variables o interruptores y LEDs según los comandos que se añadan. Además, se trata de un tipo de programación activada por eventos, es decir, no funciona en Loop, como, por ejemplo: Arduino, sino, que cada parte del código sólo se pondrá a funcionar en el momento que el simulador detecte un cambio en la variable o en el estado de un interruptor.

Esto nos permite programar en cualquier orden las líneas de código, por ejemplo, programar todas las variables juntas, y todos los interruptores en un grupo aparte, o hacerlo de manera desordenada.

Por último, se deberá guardar el archivo programado con un nombre que deberemos recordar para configurar correctamente el archivo de configuración "SIOC.ini". la extensión de dicho archivo será por defecto ".ssi" y no deberá modificarse.

Un archivo ".ssi" contiene toda la programación necesaria para funcionar un simulador, no se necesitan archivos adicionales.

Únicamente hay que cargarlo editando el archivo de configuración de "SIOC.ini" como se verá más adelante e iniciar o resetear "SIOC.exe".

A continuación, se explicará la filosofía de programación que utiliza SIOC.

2.1.1 Filosofía de la programación en SIOC

El esquema básico de cualquier programación en SIOC sigue las siguientes fases:

- **Declaraciones de variables**
- **Programar comandos**

Las declaraciones pueden ser de variables FSUIPC o de Hardware (interruptores, LEDs, etc.), quedando el esquema de la siguiente manera:

- **Declaración de la variable** FSUIPC de FSX que queramos modificar
- **Declaración de la entrada o salida** que vayamos a usar (hardware)
- **Añadir programación lógica** (comandos) a aquellas variables que queramos que, al detectarse un cambio, actúe el comando en cuestión. Normalmente con un comando IF o una función especial

Se deberá añadir la programación lógica, como secundaria de la variable que vaya a detectar un cambio en su estado, como se verá a continuación.

2.1.2 Detalle del proceso de programación

En esta sección se explicará de manera detallada la manera de proceder a la hora de programar un archivo ".ssi", tanto la declaración de variables como la explicación de los comandos y funciones.

2.1.2.1 Declaración de variables

En el editor de SIOC, se deberá hacer clic con el botón derecho en cualquier parte del código y se podrá añadir "Nueva variable" (siguiente línea) o "Insertar Variable" (línea anterior).

Podemos eliminar variables clicando con el botón derecho del ratón y seleccionando "eliminar". La tecla suprimir del teclado no tiene función alguna dentro del editor, aparte de borrar el texto seleccionado. No confundir "Variable" con "Comando". Aparecerá una ventana de parámetros **(Fig. 2.3)**. Si hacemos doble clic encima, se abrirá un menú con las opciones de la variable.

Se deberá configurar cada vez y para cada variable nueva de la manera siguiente:

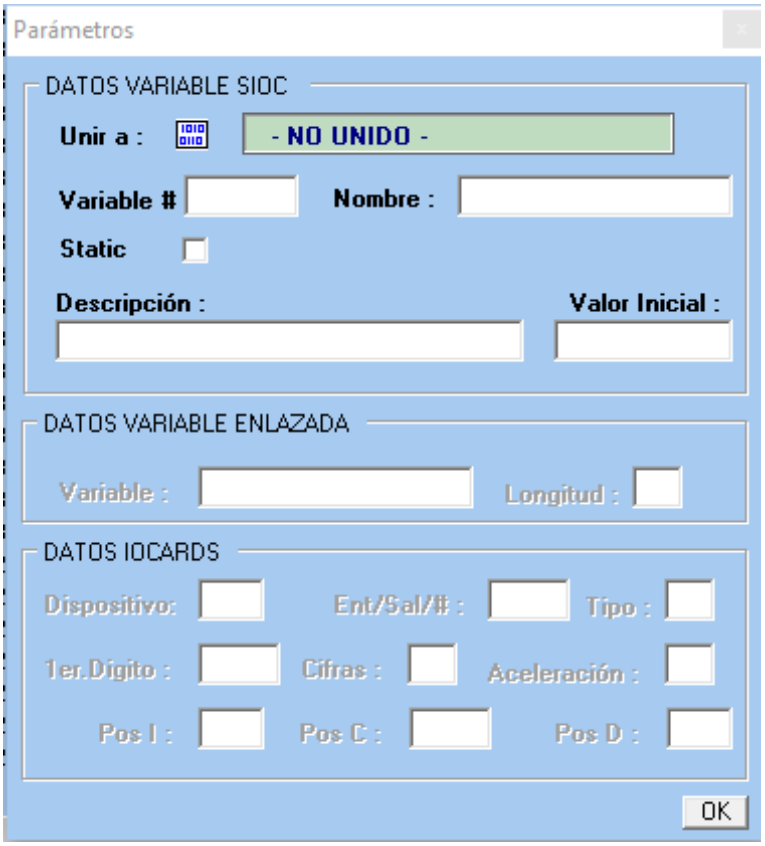

Figura 2.3: Pantalla de parámetros

El orden corresponde al mismo en el que aparecen las opciones: de arriba a abajo y de izquierda a derecha.

- **Tipo de variable (Unir a)**

Se podrá elegir si se desea declarar una variable de FSUIPC o algún elemento de hardware como por ejemplo un botón, un LED o incluso una tarjeta USB o la emulación de teclado.

Es muy importante rellenar esta opción primero para desbloquear las opciones particulares siguientes.

También, puede dejarse en blanco para crear variables vacías que sirvan como variables intermedias para mantener valores o pasar valores entre un comando u otro.

Además, se puede programar subrutinas mediante “SUBRUTINA SIOC”.

- **ID de variable (Variable #)**

Deberemos asignar un número que luego servirá para identificar la variable junto con su nombre.

Podemos asignar valores desde “1” hasta “9999”.

Se recomienda agrupar variables por grupos de números, por ejemplo: Todos los interruptores rangos de 1 a 99, todos los LEDs, 100 a 199, todas las variables FSUIPC 200 a 500, etc.
Por supuesto, puede ser cualquier valor y el ejemplo anterior es orientativo.

Se pueden desordenar los números, pero nunca repetirlos. También se pueden saltar números, no necesitan ser correlativos.

Es un campo obligatorio y necesario para identificar a posteriori todas las variables con los “programas monitor” y para programar los comandos.

- **Nombre**

Deberá ser un nombre sin espacios que permita recordar a qué corresponde la variable. Por ejemplo, si se trata de un interruptor que controla una luz, podría llamarse “intLuz” si fuera una variable que controle una luz podría ser “varLuz”.

Es importante diferenciar variables de interruptores si controlan lo mismo como se ha hecho en el ejemplo anterior, si simplemente llamamos a una variable “luz”, nunca se sabría si corresponde a una variable o a un elemento de hardware.

No puede haber dos nombres iguales.

- **Casilla “Static”**

Dejar siempre desmarcada. Bloquea la variable.

- **Descripción**

Este campo es completamente opcional, sirve para poder escribir una descripción de la declaración. Por ejemplo: “Esta variable monitoriza la altura”

- **Valor Inicial**

Este campo es opcional, aunque muchas veces es importante tenerlo en cuenta. Será el encargado de poner el valor que escribamos como valor inicial al comenzar a usar el simulador por primera vez o cada vez que pulsemos sobre “Reload” en SIOC.exe.

Por defecto, el valor inicial de las variables será el que tenga FSX al iniciar. Pero puede forzarse otro valor, por ejemplo, si queremos que se inicie con una luz encendida, pondríamos un 1 como valor inicial de la variable “varLuz”.

Si se trata de un elemento hardware es importante notar, que el valor inicial siempre será el que tenga FSX en su estado inicial o el que forcemos mediante el campo “Valor inicial”, es decir, las IOCards no son capaces de “ver” qué estado tienen los botones o interruptores en su momento de inicio, ya que sólo funciona cuando detecta cambios.

Por ello si queremos empezar una partida en FSX “Cold and Dark”, es decir, con todos los sistemas apagados, se recomienda primero apagarlos todos manualmente, guardar partida y programar todos los valores de “Valor inicial” en el archivo “.ssi” a 0.

Dependiendo del tipo de variable, si se trata de FSUIPC o de Hardware, se deberá configurar los campos explicados a continuación.

2.1.2.1.1 Variables FSUIPC

En “Unir a” seleccionaremos una de las tres opciones de “Módulo FSUIPC”.
Se podrá enviar, recibir o enviar y recibir variables a la vez.

Por regla general se usará la tercera opción cuando se declaren variables FSUIPC, para facilitar su posterior monitorización.

Seguidamente rellenaremos los campos descritos en el apartado anterior y los que se describirán a continuación (los que no aparezcan en la lista no modificarlos):

- **Variable**

Este campo es el más importante, deberá consultarse la lista de variables de FSUIPC de Peter Dowson **(Fig. 2.4)** o los anexos de Opencockpits. Se deberá copiar la dirección de memoria que corresponda a la variable que queramos programar.

30FE	2	Leading edge right outboard flap extension in degrees * 256	Ok-SimC	No
3100	1	Engine primer (just write a non-zero byte to operate the primer. This is a one-shot and reading it is meaningless)	?-SimC	?-SimC
3101	1	Alternator (1 = on, 0 = off), read for state, write to control *(This is for Alternator 1)*	?-SimC	?-SimE
3102	1	Battery (1 = on, 0 = off), read for state, write to control	?-SimC	?-SimC
3103	1	Avionics (1 = on, 0 = off), read for state, write to control	?-SimC	?-SimE
3104	1	Fuel pump (1 = on, 0 = off), read for state, write to control. For separate switches for separate fuel pumps see offset 3125. *(This is for Pump 1)*	Ok-SimC	Ok-SimE
3105	1	VOR1 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC
3106	1	VOR2 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC
3107	1	ADF1 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC

Figura 2.4: Ejemplo de localización de variable en la lista de Dowson

Por ejemplo, si queremos declarar la variable de la bomba de combustible para más adelante declarar un interruptor que la controle.

Tendremos que mirar en la lista de Dowson y veremos que la variable corresponde a la dirección de memoria 3104 (recuadro verde).

Además, nos fijamos siempre en la descripción de la variable (recuadro azul) ya que nos da las pistas importantísimas acerca de la programación de dicha variable y su funcionamiento en FSX.

Deberemos escribir la dirección de memoria siempre precediéndola de un símbolo de “dólar” de la siguiente manera: **$3104**

Además, mirando la descripción, vemos que es una variable de funcionamiento simple, es decir, con un “1” se activa y con un “0” se desactiva.

Podría haber comportamientos de variables muy complicados, como por ejemplo los de las luces o los del rumbo HDG que, los primeros, requieren una programación especial y los últimos, el uso de fórmulas matemáticas para calcular el rumbo.

- **Longitud**

30FE	2	Leading edge right outboard flap extension in degrees * 256	Ok-SimC	No
3100	1	Engine primer (just write a non-zero byte to operate the primer. This is a one-shot and reading it is meaningless)	?-SimC	?-SimC
3101	1	Alternator (1 = on, 0 = off), read for state, write to control *(This is for Alternator 1)*	?-SimC	?-SimE
3102	1	Battery (1 = on, 0 = off), read for state, write to control	?-SimC	?-SimC
3103	1	Avionics (1 = on, 0 = off), read for state, write to control	?-SimC	?-SimE
3104	1	Fuel pump (1 = on, 0 = off), read for state, write to control. For separate switches for separate fuel pumps see offset 3125. *(This is for Pump 1)*	Ok-SimC	Ok-SimE
3105	1	VOR1 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC
3106	1	VOR2 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC
3107	1	ADF1 morse ID sound (1 = on, 0 = off), read for state, write to control (see also 3122)	?-SimC	?-SimC

Figura 2.5: Localización de la longitud de la palabra de bits

En este campo se deberá copiar el número que aparece en segunda posición en la tabla de variables de Dowson.

Se trata del tamaño de bits que tiene la variable.

Normalmente para variables binarias, es decir, que actúen con un “0” o un “1”, el valor suele ser 1, aunque puede variar.

- **Tipo**

Para variables FSUIPC, normalmente se deja en blanco, pero puede ser que en la descripción de la variable de la tercera columna de las tablas de Dowson, aparezca un caso especial que obligue a poner cierto valor. Pueden ser letras o números.

Conviene recordar que, si se trata de una variable que pueda tener rangos negativos como el valor del Trim, se tendrá que especificar tipo “1” para un correcto funcionamiento.

- **Cifras**

Campo no usado, deberá dejarse en blanco.

2.1.2.1.2 Variables de Hardware

Si se desea declarar una pieza de hardware, ya sea un interruptor, botón, pulsador, encóder, potenciómetro, LED o bien, una tarjeta USB de las IOCards, se deberá seleccionar el campo correspondiente dentro de “Unir a”.

Para interruptores normales o rotativos, botones o pulsadores, se seleccionará: “Interruptor” en la sección “IOCARD MASTER” y más adelante se explicará la diferencia entre cada dispositivo y cómo diferenciarlos en el programa.

Para encóders se deberá seleccionar: “Encóders rotativos” en “IOCARD MASTER”.

Si se trata de un potenciómetro conectado a la tarjeta USBExpansion: “Entrada Analógica” dentro de “IOCARD Master”. No confundir con las posibles entradas A/D que tienen algunos dispositivos IOCards USB, que en cuyo caso deberá seleccionarse de la sección “Módulos USB”.

Los LEDs corresponderán a la sección “Salidas”.

Si se trata de alguna tarjeta USB de IOCards, se seleccionará de la sección “MODULOS USB”.

Cualquier otra tarjeta no USB, es decir, las que se pueden conectar directamente a master, se seleccionará de la sección “IOCARD MASTER”.

Una vez seleccionada se abrirán algunos o todos los campos correspondientes a la configuración específica de cada variable. (ya no se activará el campo variable ni longitud ya que son propias de las variables FSUIPC).

- **Dispositivo**

Campo que solo hay que rellenar si la declaración era de una IOCard. Normalmente se puede dejar en blanco, si hay conflicto se deberá poner el número de USB que se ha conectado la tarjeta.

- **Ent/Sal/#**

En este campo se especificará el número de la entrada o de la salida física de la IOCard, que está conectado el elemento, por ejemplo, un LED, un interruptor, un potenciómetro, etc.

- **Tipo**

Especifica el tipo de elemento en el caso de que haya que diferenciarlos.

Para interruptores (normales o rotativos) o pulsadores (On/OFF) corresponde la “I” (i latina mayúscula). En este caso cada vez que se active la variable, se mantendrá la posición que haya detectado.

Para botones momentáneos, (M-On/Off), hay que especificar un valor “P”.

Para encóders se precisa un valor “2”.

El resto no suele necesitar valores de tipo.

- **1er.Dígito**

Campo no utilizado

- **Cifras**

Campo no utilizado

- **Aceleración**

Sólo utilizado en encóders, especifica el valor de la aceleración con la que el encóder representará los valores. Normalmente pondremos 1, a no ser que queramos que al girarlo muy poco se realicen grandes cambios.

- **Pos I**

Sólo usado en potenciómetros o servos: se trata de un límite inferior, por si se quiere limitar el movimiento o indicar el valor mínimo.

- **Pos C**

Sólo usado en potenciómetros o servos: se trata de una marca central, por si se quiere modificar la curva de funcionamiento mediante la posición del punto central.

- **Pos D**

Sólo usado en potenciómetros o servos: se trata de un límite superior, por si se quiere limitar el movimiento o indicar el valor máximo.

2.1.2.1.3 Variables de emulación de teclado

La emulación de teclado permite, en lugar de modificar una variable del simulador, mandar teclas de teclado, por ejemplo, números, letras o una

combinación de ambas, incluso con caracteres especiales como mayúsculas, acentos, etc.

Se trata de una variable especial que solo se tiene que declarar una vez. No se activa ningún campo especial, únicamente habrá que rellenarse “Variable” y “Nombre” y si se desea, “Descripción”.

A la hora de querer mandar una tecla u otra, se deberá programar el comando correspondiente indicando la ID de la tecla tal y como se haya definido en el archivo “SIOC.ini”.

2.1.2.2 Comandos y funciones

Los comandos son la pieza clave de todo programa en SIOC. Se trata de la programación de la lógica y de todas las acciones que ocurren cuando pasa algún evento que hayamos definido.

Para crear un comando, como ya se ha visto anteriormente, se tendrá primero que decidir de qué declaración de variable dependerá, es decir, que se active únicamente cuando SIOC detecte un cambio en una variable.

Se hará clic derecho para abrir el menú, sobre la variable escogida. A continuación, se seleccionará “Añadir Comando”. También se puede hacer Click sobre otro comando, y en ese caso, se podrá también elegir “Insertar Comando” para ponerlo en la línea anterior. Aparecerá de nuevo un recuadro en blanco, pero esta vez desplazado a la derecha. Esto ocurre siempre con los comandos ya que indican así su pertenencia a un nivel determinado.

Seguidamente, si se hace doble clic sobre el recuadro aparecerá el menú de Parámetros **(Fig. 2.6)**, que me gusta llamar menú de Comandos y Funciones.

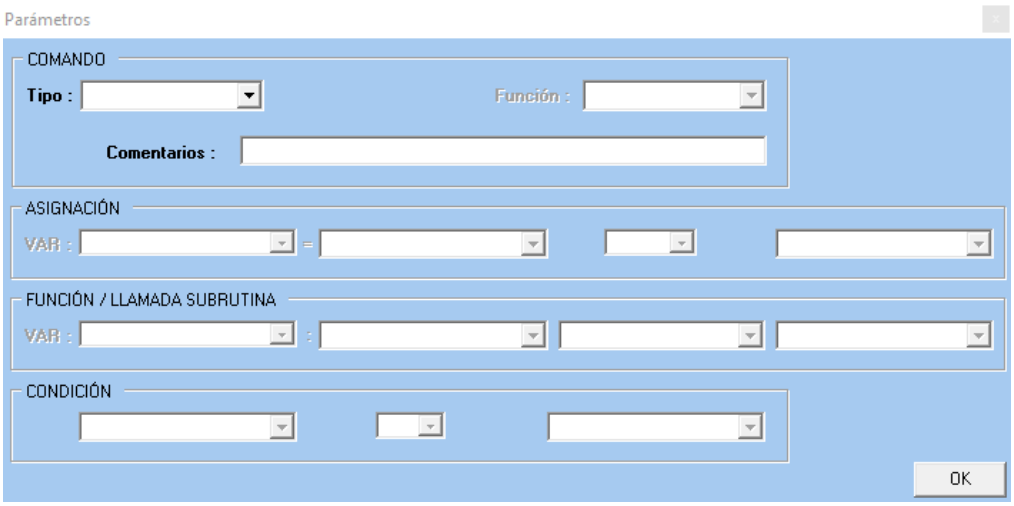

Figura 2.6: Pantalla de configuración de comandos

En dicho menú se seleccionará el tipo de comando y luego se tendrá que configurar con las opciones que se irán activando.

A continuación, se explicarán las opciones más importantes:

- **Tipo**

Menú desplegable donde encontraremos las opciones siguientes

a. Asignación

Sirve para modificar el valor de una variable. Coloquialmente sería el equivalente a decir:
"…se enciende esto" o "…se pone a valor X" etc.

Si se selecciona el tipo "Asignación" se activarán los bloques "VAR" y "=" dentro de la sección "Asignación".

VAR corresponde a la identificación de la variable que deseamos modificar y el bloque "=" corresponde al valor en sí, que podrá ser un número u el valor de otra variable.

En VAR puede usarse variables declaradas anteriormente o una serie de variables locales definidas como L0, L1 0 L2, que solo actuarán dentro del bloque de comandos de una declaración, es decir, si L0 vale 1 dentro del comando de una declaración, en otra declaración podrá usarse L0, que no valdrá 1, sino, lo que se necesite en ese caso. También existen variables locales booleanas, C0, C1 y C2, lo que puede facilitar ciertas instrucciones en comparaciones.

Si se desea pasar valores de una declaración a otra, se tendrá que crear una declaración vacía, sin unir y utilizarla como variable personal para lo que se necesite.

Un ejemplo de asignación típica, sería, por ejemplo: encender un LED cuando pulsemos un botón, se haría de la siguiente manera:

Dentro de la declaración de "BotónLed" deberíamos crear un comando, en "tipo" seleccionaríamos: "Asignación". En VAR seleccionaríamos "LED" y en "=" escribiríamos 1 para activarlo o 0 para desactivarlo. Nótese que esta programación no funcionaría ya que no hemos usado ningún comando condicional, es decir, faltaría añadir la condición IF/ELSE que se explicará más adelante.

También se podrá realizar operaciones matemáticas simples: sumar, restar, multiplicar y dividir.

Para ello, en VAR seleccionar la variable que queremos modificar. En “=” escribir el valor o la variable que se quiera adquirir un valor anterior y luego seleccionar el símbolo matemático.

Se abrirá un nuevo campo a la derecha donde podremos escribir un número o variable para realizar la operación.

Por ejemplo, queremos que la variable vacía “Resultado” que de momento vale 0, se le sume cada vez que se active el comando, 1.
Escribiríamos: “Resultado” = “Resultado” “+” “1”.

b. Función

Si queremos utilizar una función específica de las IOCards, se deberá seleccionar la opción Función y se activará la casilla correspondiente justo a su derecha.

Existen varias funciones, aunque las dos que se explicarán a continuación son las más utilizadas en el simulador de la EETAC:

Setbit/ClearBit = Modifica bits dentro de una palabra de bits de una variable, útil cuando en la descripción de una variable de la tabla de Dowson se especifica que sólo se puede actuar sobre ella mediante la modificación de esos bits, como, por ejemplo, las luces. No confundir con “changebit o changebitn”, que no sirven para ese propósito y son fuente común de errores a la hora de programar.

Rotate = Permite incrementar o decrementar valores, y además permite definir rangos de variables que vayan de un número A a un B y cuando se sobrepase B vuelvas a A y viceversa.

Muy útil al programar ciertos eventos con encóders como por ejemplo rumbos, que van de 0 a 360º y si sigues girando quieres que vuelva al 0.

Evita tener que crear una función que vaya sumando de uno en uno o restando, cada vez que detecta una posición del encóder.

c. Condición IF

Permite la creación de una condición IF, para programar lógicas. Puede añadirse varios IF uno dentro de otro. Cada IF aparecerá en un nivel nuevo. Se activarán las tres opciones inferiores.

En la primera irá la variable que queremos condicionar, en la segunda se seleccionará la operación lógica y por último el valor o la variable que queremos comparar.

d. Condición ELSE

Permite la creación de una condición ELSE, para programar lógicas. Puede añadirse varios ELSE uno dentro del otro.

e. Llamada Subrutina

Mediante esta opción se podrá llamar a una subrutina creada anteriormente. Sólo se tendrá que seleccionar cuál en la lista “VAR” y se ejecutará el código que se encuentre en la subrutina.

- **Comentarios**

Es un campo opcional donde pueden escribirse comentarios sobre la programación

2.1.2.3 Ejemplo de programación

En el siguiente ejemplo (**Fig. 2.7 y Fig. 2.8**) se verá el código tal cual debería ser programado para un caso que engloba un ejemplo real utilizado en el simulador portátil y el uso de una función especial (Setbit y Clearbit). Después se explicarán los detalles.

En la siguiente captura puede verse el código en el editor de SIOC.exe:

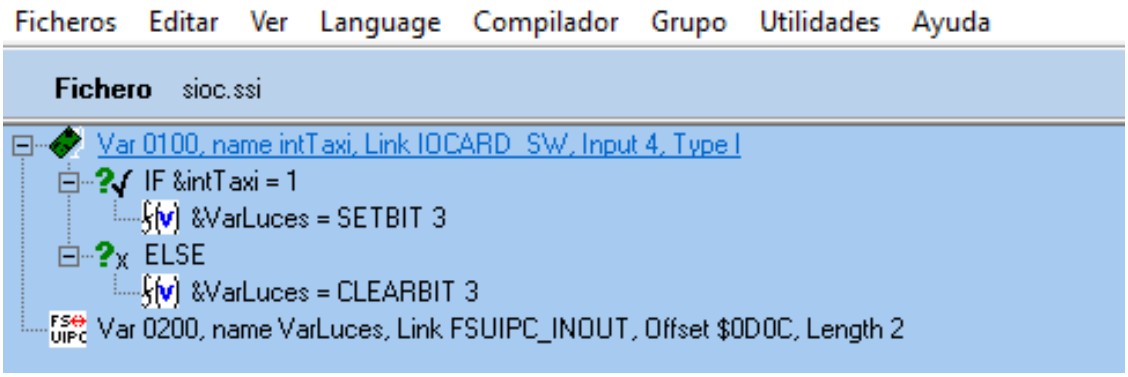

Figura 2.7: Ejemplo en editor de SIOC

Y seguidamente puede verse el mismo código una vez exportado a TXT:

```
Var 0100, name intTaxi, Link IOCARD_SW, Input 4, Type I
{
  IF &intTaxi = 1
  {
    &VarLuces = SETBIT 3
  }
  ELSE
  {
    &VarLuces = CLEARBIT 3
  }
}

Var 0200, name VarLuces, Link FSUIPC_INOUT, Offset $0D0C, Length 2
```

Figura 2.8: Ejemplo en TXT

El código del ejemplo es capaz de leer el estado de un interruptor y encender la luz de Taxi o apagarla según su posición.

En ambas imágenes aparecen cada uno de los parámetros de configuración de las variables y de los comandos.

Además, puede notarse que es posible la programación desde la interfaz gráfica como en TXT utilizando programación clásica.

Para ello primero se ha declarado el interruptor, que en este caso está conectado a la entrada “4” y es de tipo “I” (“i” latina), ya que se trata de un interruptor ON-OFF.

Seguidamente, se ha declarado la variable que controla el grupo de luces. Se trata de una variable especial que, según la descripción de las tablas de Dowson, para hacerla funcionar se tiene que cambiar el valor del Bit correspondiente a cada luz.

Puede verse que el bit correspondiente a la luz de Taxi corresponde al número “3”. Por lo tanto, para su activación, se usará el comando “Función” y se seleccionará “Setbit 3” para activar el bit 3 y “Clearbit 3” para desactivarlo.

Podemos ver que la dirección de la variable corresponde a la $0D0C **(Fig. 2.9)** y tiene una longitud de 2 bits.

0D0C	2	Lights, a switch for each one (bits from lo to hi): 0 Navigation 1 Beacon 2 Landing 3 Taxi 4 Strobes 5 Instruments 6 Recognition 7 Wing 8 Logo 9 Cabin	Ok-SimC	Ok-SimE *(Intl decode)*

Figura 2.9: Ejemplo de variable de luces. Lista de Peter Dowson

Como interesa modificar el estado de las luces cuando se activa el interruptor, y no al revés, la programación lógica (comandos) se ha incluido dentro de la declaración del interruptor y no de la variable.

En todos los casos de realizar alguna acción mediante interruptores, se deberá utilizar siempre los comandos IF y ELSE ya que, de otra manera, el programa no sabrá cómo actuar.

2.2 Configuración "SIOC.ini"

El archivo de configuración SIOC.ini, es el encargado de gestionar la configuración inicial de todo el sistema IOCards.

Podrá encontrarse toda la configuración necesaria para el simulador portátil en el (**Anexo 2. Configuración de "sioc.ini"**)

Es necesario conocerlo y modificarlo para su correcto funcionamiento una única vez a no ser que se modifique el número o tipo de tarjetas IOCards, o se desee utilizar un archivo ".ssi" de programación diferente. Se encuentra en la carpeta de instalación de SIOC, por defecto en:

C:\Program Files (x86)\IOCards\SIOC

Este archivo puede abrirse mediante el bloc de notas, o directamente desde el botón "Edit.ini" **(Fig. 2.10)** de "SIOC.exe".

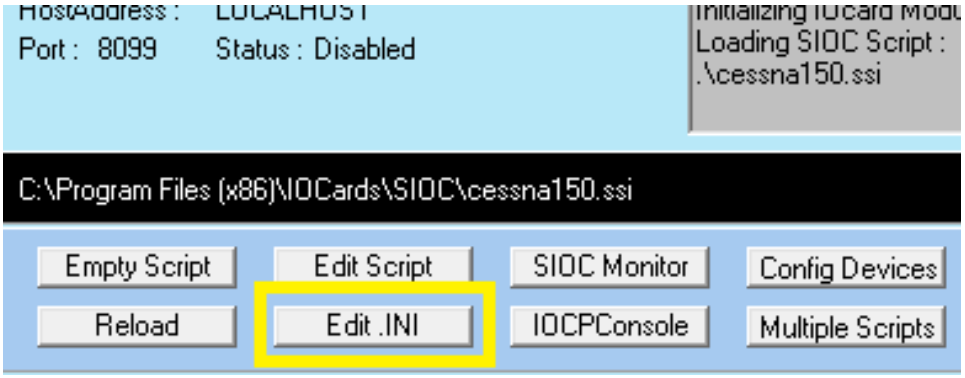

Figura 2.10: Localización de "Edit.ini" para configurar "Sioc.ini"

El archivo SIOC.ini deberá ser modificado únicamente por personas con los conocimientos necesarios, ya que su incorrecto uso puede dar lugar a un mal funcionamiento de las tarjetas IOCards.

Consta de varias secciones:

- Configuración de SIOC
- Dispositivos virtuales
- Módulo de monitorización
- Módulo de IOCards
- Configuración de IOCards
  - Master
  - Otras tarjetas
- Módulo FSUIPC
- Módulos cliente IOCP (varios)
- Módulo de sonido
- Emulación del teclado

De estos, los más importantes, y los que se verá en detalle en el siguiente tema son:

- **Configuración de SIOC**
- **Configuración de IOCards**
  - Master
  - Otras tarjetas
- **Emulación del teclado**

El resto de secciones controlan aspectos muy específicos, no necesarios para las funciones normales de un simulador, además, otras no tienen que ser

tocadas, ya que están perfectamente calibradas para un correcto uso de las tarjetas, IOCards, por lo tanto, únicamente se modificará, si fuera necesario lo explicado a continuación.

2.2.1 Sección “Configuración de SIOC”

Se deberá añadir a la línea inferior: CONFIG_FILE.|\ el nombre exacto del archivo de programación “.ssi” que hayamos creado mediante el editor como se ha explicado anteriormente.

Por ejemplo, si hemos creado una programación para un simulador del avión acrobático Extra 300, y lo hemos llamado extra300.ssi, deberá quedar: CONFIG_FILE.|\extra300.ssi

El resto de opciones NO HAY que cambiarlas.

En el caso del simulador portátil, el archivo se llamará “**portatil.ssi**”

2.2.2 Sección “Configuración de IOCards Master”

En este apartado se encuentra la configuración de la tarjeta principal.

El resto de líneas que aparecen en esta sección están bajo corchetes, con lo que son meras descripciones, pueden ser borradas si se desea.

La línea más importante es la que aparece SIN los corchetes:

MASTER=X,X,X,X

Aunque la función se llame “MASTER”, como se utiliza la tarjeta USBExpansion, realmente, la configuración de esta línea, se trata de la configuración de la USBExpansion y no de la Master.

Cada X representa un valor que se explicará a continuación:

1. *El primer valor corresponde a la **ID de prioridades**. Corresponde una ID por cada USBExpansion conectada al mismo PC.*

Si sólo hay una tarjeta conectada (independientemente de las Master que haya), la ID será 0.

Si hay más de una deberemos crear otra línea igual, la primera con un 0 y la segunda con un 1, quedando así:

MASTER=0,X,X,X.

Ejemplo con dos USBExpansion conectadas:

MASTER=0,X,X,X
MASTER=1,X,X,X

2. *El Segundo valor corresponde al **tipo de tarjeta** que conectamos la Master. Para nuestro caso siempre se trata del número 4 ya que se conectarán mediante la tarjeta USBExpansión.* En las descripciones del archivo "SIOC.ini", entre corchetes puede verse cada valor a qué tipo corresponde cada valor.

3. *El tercer valor corresponde al **número de tarjetas máster conectadas**. Si es una, será 1, dos: 2, etc.*

Ejemplo con dos USBExpansion conectadas, la primera con 4 master y la segunda con 2:

MASTER=0,4,4,X
MASTER=1,4,2,X

4. *El último valor es el **Device number** puede dejarse a 0 o hacer que corresponda al número del puerto USB conectado.*

Ejemplo: dos USBExpansion conectadas, la primera con 4 master y la segunda con 2, además las USBExpansion están conectadas al puerto 22 y 26 respectivamente.

MASTER=0,4,4,22 (o bien 0,4,4,0 para asignar la ND automáticamente)
MASTER=1,4,2,25 (o bien 1,4,2,0 para asignar la ND automáticamente)

2.2.3 Configuración de IOCards "Otras tarjetas"

En este campo se deberá declarar cada una de las tarjetas USB que se desee utilizar, añadiendo las líneas correspondientes. Sólo se deberá indicar el ND y la ID. Normalmente lo dejaremos como (0,0) a no ser que se usen más de una tarjeta iguales.

Si la programación no funciona, se recomienda cambiar el segundo “0” por el número de USB conectado, por ejemplo: 22, 25, o el que se haya asignado en ese momento.

Las posibles tarjetas que se pueden declarar son las siguientes:

USBStepper, USBKeys, USBServos, USBRelays, USBDCmotor

Aunque en el caso del simulador portátil diseñado, no se hará uso de ninguna de ellas, por lo tanto, no se necesitará incluirlas.

2.2.4 Sección “Emulación del teclado”

Esta sección corresponde a un “diccionario” de asignaciones de teclas. A cada valor de la variable “Emulador de teclado” podemos asignarle (o dejarlo por defecto), un valor de teclado o una combinación de teclas.

Todos los valores de variables comienzan por una almohadilla, seguida del valor, un signo igual y la tecla en mayúscula del teclado que queramos emular.

Las teclas siempre se envían por defecto en minúscula, aunque se escriban en mayúscula. Para enviar una tecla mayúscula, se deberá usar la combinación \3 y \4, que corresponden a presionar la tecla “Mayús” y dejar de presionar la misma tecla.

Por ejemplo, queremos enviar una “E” en mayúscula: \3E\4
Significa literalmente: <<presiona mayúscula, luego la “e” y luego deja de presionar mayúscula>>

Todas las combinaciones posibles se encuentran documentadas en los manuales de la tarjeta “USBKeys”.

2.3 Mantenimiento mediante software

La idea de incluir un apartado de mantenimiento mediante software, es la de poder dar unas pequeñas pautas basadas en mi propia experiencia, para poder solucionar pequeños errores que pueden surgir con el paso del tiempo.

El mantenimiento del simulador portátil pasará por un correcto uso del mismo y la revisión periódica de sus componentes para asegurar que estén en buen estado.

Además, para poder detectar fallos tanto en hardware como en software, se mostrarán y explicarán dos herramientas basadas en software muy útiles.

2.3.1 Monitorización con SIOCMonitor

SIOCMonitor es un programa que permite conocer el estado de todas las entradas o salidas que tiene cualquier tarjeta IOCard conectada al PC.

Se accede mediante SIOC.exe como puede verse en la siguiente imagen **(Fig. 2.11)**. Se tiene que hacer doble clic encima de la tarjeta que queramos monitorizar.

Cada Tarjeta USB tendrá su propio tipo de monitorización.

En esta sección únicamente se explicará la correspondiente a la USBExpansion, que es la que se utiliza en el simulador de la EETAC y se utilizará en el simulador portátil.

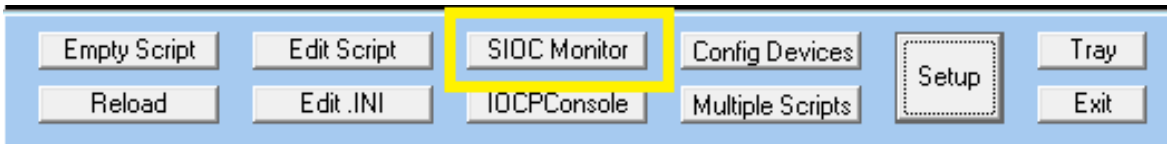

Figura 2.11: Acceso a SIOC monitor mediante "Sioc.exe"

Con este programa se podrá ver fácilmente si algún interruptor, botón, led, Display o potenciómetro está fallando, ya que el programa permite encender una señal luminosa en cada elemento que pulsemos físicamente y al revés, dar la orden para que se enciendan a la vez todos los LEDs o displays.

Además, permite enviar órdenes individuales para probar cada elemento por separado.

También es útil para conocer el número de la entrada a la que está conectado cada elemento.

En la imagen siguiente **(Fig. 2.12)** se podrá ver una captura del programa y la explicación de cada sección del mismo a continuación de la misma.

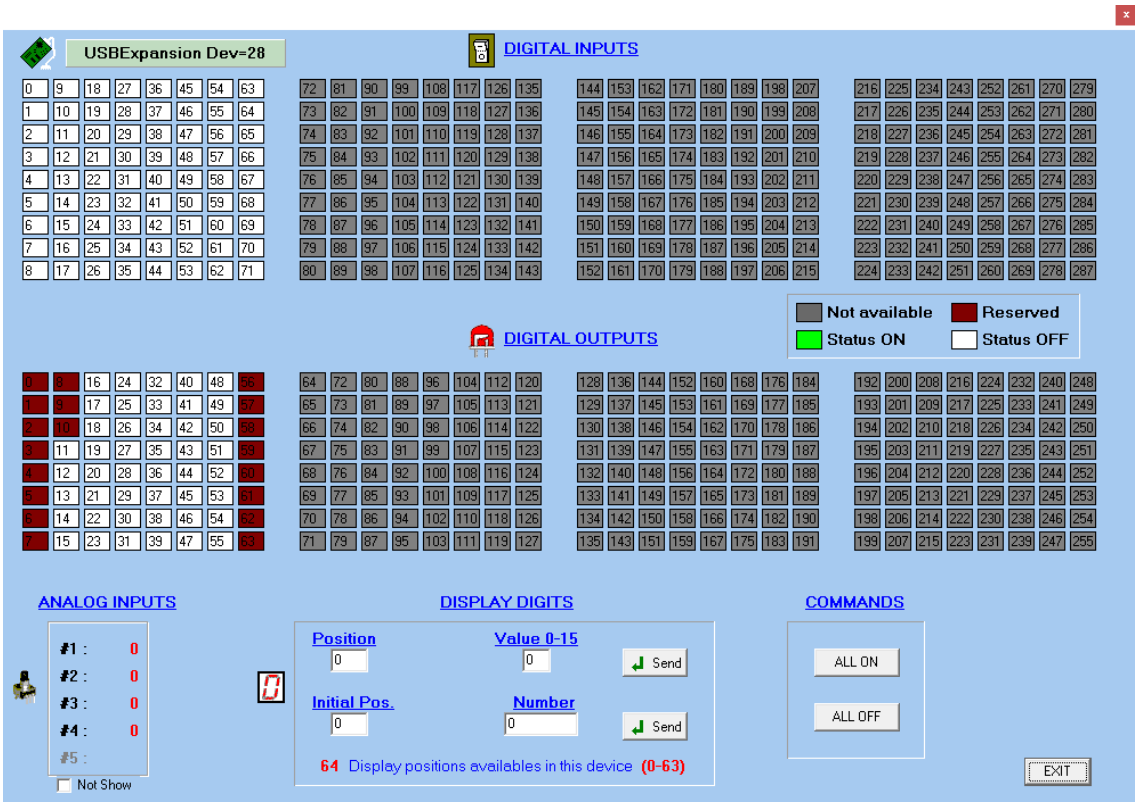

Figura 2.12: Pantalla de SIOC Monitor

En la sección superior “Digital inputs” se puede comprobar cada entrada que conectemos si funciona correctamente. Para ello pulsaremos físicamente sobre los botones de nuestro panel (botones, encóders, interruptores, etc.) y si funcionan correctamente, se iluminará de color verde en el momento que detecte la posición de “activado” o “1”. Si no se activa es posible que la conexión falle.

La segunda sección corresponde a las salidas de LEDs. Podremos presionar sobre el icono de abajo a la derecha “Commands > All on” para encender de golpe todos los leds o apagarlos mediante “All off”. Así se podrá comprobar su correcto funcionamiento. Además, encenderá o apagará todos los Displays, lo cual también nos servirá para ver su funcionamiento.

La sección inferior izquierda “Analog inputs” corresponde a las lecturas de los potenciómetros de las entradas A/D de la USBExpansion. Un correcto funcionamiento nos mostrará un único valor para cada posición.

Por último, la sección inferior central “Display Digits” nos permite enviar números a un determinado Display. Lo cual no es necesario ya que podemos utilizar, como he explicado antes, la función “All on” para encender todos los outputs incluidos displays.

No se necesita tener FSX encendido. Sólo se tiene que tener la tarjeta conectada y SIOC.exe ejecutándose.

2.3.2 Monitorización con IOCPConsole

IOCPConsole (**Fig. 2.14**) permite conocer a tiempo real el valor de las variables que se hayan programado ya sean de FSUIPC o declaraciones de hardware como potenciómetros, encóders o interruptores.

Podemos acceder a IOCPConsole desde SIOC.exe **(Fig. 2.13)**

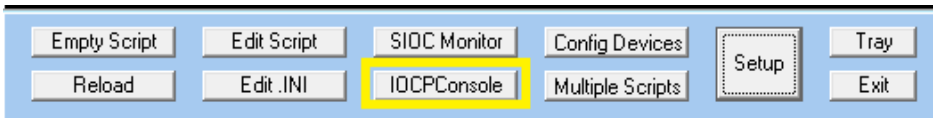

Figura 2.13: Acceso a IOCPConsole mediante "Sioc.exe**"**

Para que funcione correctamente se tiene que utilizar con FSX encendido y en segundo plano.

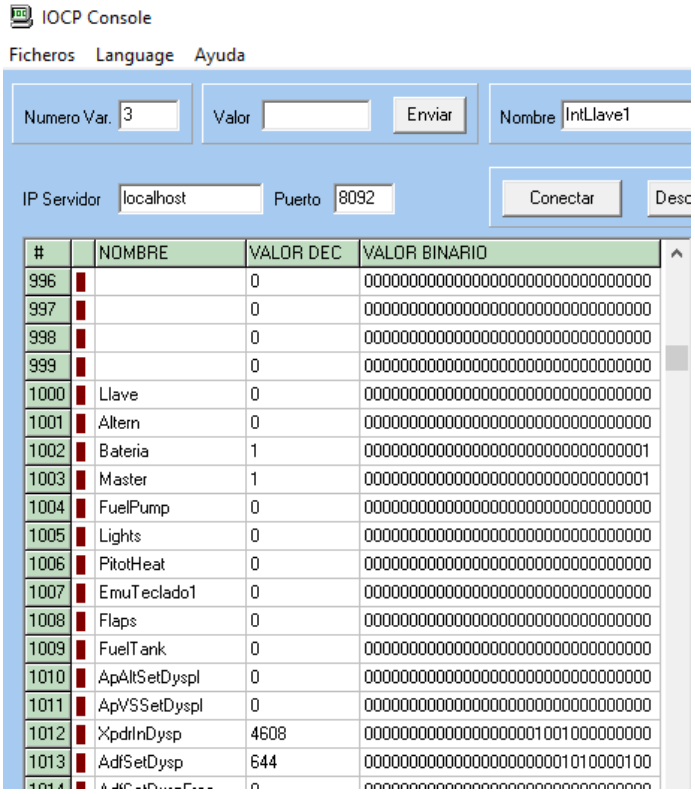

Figura 2.14: Vista de IOCPConsole

Para hacerlo funcionar se deberá clicar al botón “Conectar”.

Aparecerá en la primera columna, la lista de todas las declaraciones que hayamos hecho y su valor en la segunda columna.

La tercera columna corresponde al valor binario que no necesitaremos.

Las variables se encuentran ordenadas por el número que se les haya asignado en el momento de la programación (campo **Variable #**).

Como puede deducirse, es muy útil para hacer simulaciones de programación ya que no necesitaremos tener hardware para ver qué pasaría si ocurre algún evento programado.

Es posible emular hardware mediante el campo “Enviar” donde podremos cambiar valores de variables manualmente y ver qué ocurre en la segunda columna.

En la imagen se ha dejado ver a propósito el estado de botones de FSX. En “IOCPConsole” puede verse cómo aquellos que están en On, se encuentran con un “1” en la segunda columna tal y como se esperaba.

CONCLUSIONES

En este trabajo se ha propuesto el diseño de un simulador de vuelo portátil para la Universidad Politécnica de Cataluña.

Dicho diseño ha constado de una parte hardware y una parte software.

La parte de hardware se ha diseñado teniendo en cuenta los objetivos propuestos por el profesorado: que sea fácilmente transportable y con un presupuesto no superior a 3000€. Objetivos logrados mediante la elección del sistema más económico y versátil de tarjetas IOCards para controlar los aspectos de hardware del simulador, y una elección del resto de componentes inteligente y respetuosa con el medio ambiente, teniendo en cuenta los requisitos y prestaciones necesarios.

Tal y como se pedía, el diseño propuesto permite su fácil transporte gracias a su tamaño compacto y compartimentos para guardar el teclado. Además, se ha diseñado con una disposición de elementos y una elección de colores y detalles muy realistas y ha podido personalizarse y añadirse el logotipo de la escuela, para poder realizar tareas de promoción. Este proceso de diseño gráfico ha precisado conocimientos de diseño 2D, 3D, y temas de aeronáutica para saber la disposición de los instrumentos y sus funciones.

Como se ha visto, se ha tenido que aprender a utilizar el sistema de tarjetas IOCards y aplicar conocimientos de electrónica y soldadura.

Comparando el diseño propuesto con otras opciones del mercado, se ha visto que, realizando el primero, los costes finales se han reducido aproximadamente en 3000€ respecto al simulador de vuelo completo más barato.

La parte de software ha precisado del estudio de un nuevo lenguaje de programación exclusivo de simulación aérea: lenguaje SIOC, el cual ha permitido crear un simulador de vuelo perfectamente funcional y muy completo. Para ello se ha necesitado conocimientos de informática, telecomunicaciones y aeronáutica (en el apartado de las radios de los aviones y los instrumentos de vuelo en general) y soltura en programación de nuevos lenguajes.

Como puede verse, ha sido un proyecto muy completo que ha tenido que tener en cuenta una gran parte de las asignaturas de la carrera y que ha precisado de mucho tiempo de autoaprendizaje y pruebas para poder realizarlo con éxito.

Creo en la factibilidad de este diseño dada la experiencia adquirida con los años de estudio que he realizado desde 2013 sabiendo que quería hacer mi TFG relacionado con la simulación. Además, la seguridad de saber que podré afrontar el reto del montaje, una vez se disponga de las piezas, me la ha dado la construcción y el mantenimiento de dos simuladores personales y el actual simulador de la Universidad.

REFERENCIAS

[1] http://www.cealweb.net/IPBCEAL/index.php?/page/index.html/_/noticias/simulacion-aerea/barcelona-air-sim-meeting-2013-r121

[2] http://www.aeronauticscat.org/

[3] http://www.opencockpits.com/index.php/es/iocards/item/descripcion?category_id=63

[4] http://www.lekseecon.nl/downloads/How%20to%20SIOC.pdf [Enlace a descarga de PDF]

[5] https://www.youtube.com/watch?v=Y9Wr4vdw7sM&list=PLGltNDsmkqccW5uUgawP5fiRt7DNxgGWD

[6] https://www.youtube.com/watch?v=932cpc1r47U&feature=youtu.be

[7] http://www.semana.com/nacion/multimedia/escuela-de-aviacion-policial-los-guardianes-del-cielo/432755-3

[8] https://www.logitech.com/es-es/product/wireless-touch-keyboard-k400-plus

[9] http://www.simioboard.com/

[10] https://www.arduino.cc/

[11] https://buy.garmin.com/en-SG/digital/p/115

[12] http://www.saitek.com/uk/prod-bak/yoke.html

[13] https://www.ivao.aero/

[14] https://www.sketchup.com/es

[15] http://www.opencockpits.com/index.php/es/descargas/category/iocards

[16] https://www.wilcopub.com/prop-cockpit-trainer.html

[17] https://www.prepar3d.com/

[18] http://www.opencockpits.com/index.php/en/download/item/sioc-ver-50b5?category_id=39

[19] http://www.schiratti.com/dowson.html

ANEXOS

1. Código SIOC

A continuación, se incluirá todo el código de la programación del simulador, para ser importado como TXT en SIOC. Cabe notar que los primeros campos, en los que pone “By Manuel Velez” es un campo que se genera automáticamente siempre que se exporta cualquier código SIOC a TXT, independientemente de quién lo haya programado, se trata de los créditos del creador de las tarjetas, y la programación ha sido realizada por mí, utilizando parte del código que creé en el simulador actual de la escuela y en mis propios simuladores de vuelos, incluyendo código nuevo específico para el simulador portátil.

```
//
************************************************************
**************
// * Config_SIOC ver 4.5 - By Manuel Velez - www.opencockpits.com
//
************************************************************
**************
// * FileName : 150.txt
// * Date : 24/01/2018



Var 1007, name EmuTeclado1, Link KEYS

Var 1008, name Starter, Link FSUIPC_OUT, Offset $0892, Length 2

Var 0107, name RotStarterOff, Link IOCARD_SW, Input 14, Type I
{
 IF &RotStarterOff = 1
 {
  &Starter = 0
 }
}

Var 0108, name RotStarteRight, Link IOCARD_SW, Input 13, Type I
{
 IF &RotStarteRight = 1
 {
  &Starter = 1
 }
}

Var 0109, name RotStarterLeft, Link IOCARD_SW, Input 12, Type I
{
 IF &RotStarterLeft = 1
 {
  &Starter = 2
 }
}

Var 0110, name RotStarterBoth, Link IOCARD_SW, Input 11, Type I
{
 IF &RotStarterBoth = 1
 {
  &Starter = 3
 }
}

Var 0111, name RotStarterSta, Link IOCARD_SW, Input 10, Type I
{
 IF &RotStarterSta = 1
 {
  &Starter = 4
 }
}

Var 1001, name Altern, Link FSUIPC_OUT, Offset $3101, Length 1, Value 0

Var 0004, name IntAlt, Link IOCARD_SW, Input 14, Type I
{
 IF &IntAlt = 1
 {
  &Altern = 1
 }
 ELSE
 {
  &Altern = 0
 }
}

Var 1002, name Bateria, Link FSUIPC_INOUT, Offset $3102, Length 1, Value 0
{
 CALL &LedAPSub, &Master
 CALL &FlapsSub, &Master
 CALL &AudioPanelSub, &Master
 CALL &DisplaysSub, &Bateria
}

Var 0005, name IntBat, Link IOCARD_SW, Input 15, Type I
{
 IF &IntBat = 1
 {
  &Bateria = 1
 }
 ELSE
 {
  &Bateria = 0
 }
}
```

```
Var 1003, name Master, Link FSUIPC_INOUT, Offset $3103, Length 1, Value 0
{
 CALL &LedAPSub, &Master
 CALL &FlapsSub, &Master
 CALL &AudioPanelSub, &Master
 CALL &DisplaysSub, &Master
}

Var 0006, name IntMaster, Link IOCARD_SW, Input 16, Type I
{
 IF &IntMaster = 1
 {
  &Master = 1
 }
 ELSE
 {
  &Master = 0
  &PitchServo = 512
  &BankServo = 512
 }
}

Var 1004, name FuelPump, Link FSUIPC_OUT, Offset $3104, Length 1, Value 0

Var 0007, name IntFuelPump, Link IOCARD_SW, Input 0, Type I
{
 IF &IntFuelPump = 1
 {
  &FuelPump = 1
 }
 ELSE
 {
  &FuelPump = 0
 }
}

Var 1005, name Lights, Link FSUIPC_OUT, Offset $0D0C, Length 2, Value 0

Var 0008, name IntBcnLights, Link IOCARD_SW, Input 1, Type I
{
 IF &IntBcnLights = 1
 {
  &Lights = SETBIT 1
 }
 ELSE
 {
  &Lights = CLEARBIT 1
 }
}

Var 0009, name IntLandLights, Link IOCARD_SW, Input 2, Type I
{
 IF &IntLandLights = 1
 {
  &Lights = SETBIT 2
 }
 ELSE
 {
  &Lights = CLEARBIT 2
 }
}

Var 0010, name IntTaxiLights, Link IOCARD_SW, Input 3, Type I
{
 IF &IntTaxiLights = 1
 {
  &Lights = SETBIT 3
 }
 ELSE
 {
  &Lights = CLEARBIT 3
 }
}

Var 0011, name IntNavLights, Link IOCARD_SW, Input 4, Type I
{
 IF &IntNavLights = 1
 {
  &Lights = SETBIT 0
 }
 ELSE
 {
  &Lights = CLEARBIT 0
 }
}

Var 0012, name IntStrobeLight, Link IOCARD_SW, Input 5, Type I
{
 IF &IntStrobeLight = 1
 {
  &Lights = SETBIT 4
 }
 ELSE
 {
  &Lights = CLEARBIT 4
 }
}

Var 1006, name PitotHeat, Link FSUIPC_OUT, Offset $029C, Length 1, Value 0

Var 0013, name IntPitot, Link IOCARD_SW, Input 6, Type I
{
 IF &IntPitot = 1
 {
  &PitotHeat = 1
 }
 ELSE
 {
  &PitotHeat = 0
 }
}

Var 2595, name FuelValve, Link FSUIPC_INOUT, Offset $3590, Length 1, Value 1

Var 0014, name IntFuelValve, Link IOCARD_SW, Input 18, Type I
{
 IF &IntFuelValve = 1
 {
  &FuelValve = 1
 }
 ELSE
 {
  &FuelValve = 0
 }
}

Var 1508, name Flaps, Link FSUIPC_INOUT, Offset $0BDC, Length 4, Value 0
{
 CALL &FlapsSub, &Flaps
}

Var 3000, name FlapsSub, Link SUBRUTINE
{
 IF &Master = 1
 {
  IF &Bateria = 1
  {
```

```
    IF &Flaps = 0
    {
     &LedFlap0 = 1
    }
    ELSE
    {
     &LedFlap0 = 0
    }
    IF &Flaps = 5461
    {
     &LedFlap10 = 1
    }
    ELSE
    {
     &LedFlap10 = 0
    }
    IF &Flaps = 10922
    {
     &LedFlap20 = 1
    }
    ELSE
    {
     &LedFlap20 = 0
    }
    IF &Flaps = 16383
    {
     &LedFlapFull = 1
    }
    ELSE
    {
     &LedFlapFull = 0
    }
   }
   ELSE
   {
    &LedFlap0 = 0
    &LedFlap10 = 0
    &LedFlap20 = 0
    &LedFlapFull = 0
   }
  }
  ELSE
  {
   &LedFlap0 = 0
   &LedFlap10 = 0
   &LedFlap20 = 0
   &LedFlapFull = 0
  }
}

Var 0015, name IntFlapsDown, Link IOCARD_SW, Input 20, Type I
{
 IF &IntFlapsDown = 1
 {
  IF &Flaps <= 10922
  {
   &Flaps = &Flaps + 5461
  }
 }
 CALL &FlapsSub, &Flaps
}

Var 0016, name IntFlapsUp, Link IOCARD_SW, Input 19, Type I
{
 IF &IntFlapsUp = 1
 {
  IF &Flaps >= 5461
  {
   &Flaps = &Flaps - 5461
  }
 }
 CALL &FlapsSub, &Flaps
}

Var 1009, name FuelTank, Link FSUIPC_OUT, Offset $0AF8, Length 2

Var 0017, name IntTankLeft, Link IOCARD_SW, Input 21, Type I
{
 IF &IntTankLeft = 1
 {
  &FuelTank = 2
 }
}

Var 0018, name IntTankBoth, Link IOCARD_SW, Input 22, Type I
{
 IF &IntTankBoth = 1
 {
  &FuelTank = 1
 }
}

Var 0019, name IntTankRight, Link IOCARD_SW, Input 23, Type I
{
 IF &IntTankRight = 1
 {
  &FuelTank = 3
 }
}

Var 1017, name ApApVar, Link FSUIPC_IN, Offset $07BC, Length 4
{
 CALL &LedAPSub, &ApApVar
}

Var 1018, name ApHdgVar, Link FSUIPC_IN, Offset $07C8, Length 4
{
 CALL &LedAPSub, &ApHdgVar
}

Var 1019, name ApNavVar, Link FSUIPC_IN, Offset $07C4, Length 4
{
 CALL &LedAPSub, &ApNavVar
}

Var 1020, name ApAprVar, Link FSUIPC_IN, Offset $0800, Length 4
{
 CALL &LedAPSub, &ApAprVar
}

Var 1021, name ApRevVar, Link FSUIPC_IN, Offset $0804, Length 4
{
 CALL &LedAPSub, &ApRevVar
}

Var 1022, name ApAltVar, Link FSUIPC_IN, Offset $07D0, Length 4
{
 CALL &LedAPSub, &ApAltVar
}

Var 3001, name LedAPSub, Link SUBRUTINE
{
 IF &Master = 1
 {
  IF &Bateria = 1
  {
   IF &ApApVar = 1
   {
    &LedApAp = 1
   }
```

```
    ELSE
    {
     &LedApAp = 0
    }
    IF &ApHdgVar = 1
    {
     &LedApHdg = 1
    }
    ELSE
    {
     &LedApHdg = 0
    }
    IF &ApNavVar = 1
    {
     &LedApNav = 1
    }
    ELSE
    {
     &LedApNav = 0
    }
    IF &ApAprVar = 1
    {
     &LedApApr = 1
    }
    ELSE
    {
     &LedApApr = 0
    }
    IF &ApRevVar = 1
    {
     &LedApRev = 1
    }
    ELSE
    {
     &LedApRev = 0
    }
    IF &ApAltVar = 1
    {
     &LedApAlt = 1
    }
    ELSE
    {
     &LedApAlt = 0
    }
   }
   ELSE
   {
    &LedApAp = 0
    &LedApHdg = 0
    &LedApNav = 0
    &LedApApr = 0
    &LedApRev = 0
    &LedApAlt = 0
   }
  }
  ELSE
  {
   &LedApAp = 0
   &LedApHdg = 0
   &LedApNav = 0
   &LedApApr = 0
   &LedApRev = 0
   &LedApAlt = 0
  }
}

Var 0021, name BotApOn, Link IOCARD_SW, Input
27, Type I
{
 IF &BotApOn = 1
 {
  &EmuTeclado1 = 2
  &EmuTeclado1 = 0
 }
}

Var 0022, name BotApHdg, Link IOCARD_SW, Input
28, Type I
{
 IF &BotApHdg = 1
 {
  &EmuTeclado1 = 3
  &EmuTeclado1 = 0
 }
}

Var 0023, name BotApNav, Link IOCARD_SW, Input
29, Type I
{
 IF &BotApNav = 1
 {
  &EmuTeclado1 = 4
  &EmuTeclado1 = 0
 }
}

Var 0024, name BotApApr, Link IOCARD_SW, Input
30, Type I
{
 IF &BotApApr = 1
 {
  &EmuTeclado1 = 5
  &EmuTeclado1 = 0
 }
}

Var 0025, name BotApRev, Link IOCARD_SW, Input
31, Type I
{
 IF &BotApRev = 1
 {
  &EmuTeclado1 = 6
  &EmuTeclado1 = 0
 }
}

Var 0026, name BotApAlt, Link IOCARD_SW, Input
32, Type I
{
 IF &BotApAlt = 1
 {
  &EmuTeclado1 = 7
  &EmuTeclado1 = 0
 }
}

Var 0028, name IntApAltVs, Link IOCARD_SW, Input
45, Type I

Var 1010, name ApAltSetDyspl, Link FSUIPC_INOUT,
Offset $07D4, Length 4
{
 CALL &DisplaysSub, &ApAltSetDyspl
}

Var 1011, name ApVSSetDyspl, Link FSUIPC_INOUT,
Offset $07F2, Length 2, Type 1
{
 CALL &DisplaysSub, &ApVSSetDyspl
}

Var 0029, name BotApUp, Link IOCARD_SW, Input
34, Type P
{
 CALL &atcUpDwn
 IF &IntApAltVs = 1
 {
  IF &ApAltSetDyspl <= 1993542144
  {
   L0 = 30.47999573 * 65536
   &ApAltSetDyspl = &ApAltSetDyspl + L0
  }
```

```
 }
 ELSE    // Si es VS
 {
  IF &ApVSSetDyspl <= 9800
  {
   &ApVSSetDyspl = &ApVSSetDyspl + 100
  }
 }
}

Var 0027, name BotApDown, Link IOCARD_SW, Input
33, Type P
{
 CALL &atcUpDwn
 IF &IntApAltVs = 1
 {
  IF &ApAltSetDyspl >= 1997537
  {
   L0 = 30.47999573 * 65536
   &ApAltSetDyspl = &ApAltSetDyspl - L0
  }
 }
 ELSE    // Si es VS
 {
  IF &ApVSSetDyspl >= -9800
  {
   &ApVSSetDyspl = &ApVSSetDyspl - 100
  }
 }
}

Var 6301, name atcUpDwn, Link SUBRUTINE
{
 IF &BotApUp = 1
 {
  IF &BotApDown = 1
  {
   IF &tempATC = 0
   {
    &tempATC = 1
   }
   ELSE
   {
    &tempATC = 0
   }
  }
 }
}

Var 0030, name BotXpdr0, Link IOCARD_SW, Input
36, Type I
{
 CALL &XpdrSub, &BotXpdr0
 IF &BotXpdr0 = 1
 {
  &EmuTeclado1 = 36
  &EmuTeclado1 = 0
 }
}

Var 0031, name BotXpdr1, Link IOCARD_SW, Input
37, Type I
{
 CALL &XpdrSub, &BotXpdr1
 IF &BotXpdr1 = 1
 {
  &EmuTeclado1 = 27
  &EmuTeclado1 = 0
 }
}

Var 0032, name BotXpdr2, Link IOCARD_SW, Input
38, Type I
{
 CALL &XpdrSub, &BotXpdr2
 IF &BotXpdr2 = 1
 {
  &EmuTeclado1 = 28
  &EmuTeclado1 = 0
 }
}

Var 0033, name BotXpdr3, Link IOCARD_SW, Input
39, Type I
{
 CALL &XpdrSub, &BotXpdr3
 IF &BotXpdr3 = 1
 {
  &EmuTeclado1 = 29
  &EmuTeclado1 = 0
 }
}

Var 0034, name BotXpdr4, Link IOCARD_SW, Input
40, Type I
{
 CALL &XpdrSub, &BotXpdr4
 IF &BotXpdr4 = 1
 {
  &EmuTeclado1 = 30
  &EmuTeclado1 = 0
 }
}

Var 0035, name BotXpdr5, Link IOCARD_SW, Input
41, Type I
{
 CALL &XpdrSub, &BotXpdr5
 IF &BotXpdr5 = 1
 {
  &EmuTeclado1 = 31
  &EmuTeclado1 = 0
 }
}

Var 0036, name BotXpdr6, Link IOCARD_SW, Input
42, Type I
{
 CALL &XpdrSub, &BotXpdr6
 IF &BotXpdr6 = 1
 {
  &EmuTeclado1 = 32
  &EmuTeclado1 = 0
 }
}

Var 0037, name BotXpdr7, Link IOCARD_SW, Input
43, Type I
{
 CALL &XpdrSub, &BotXpdr7
 IF &BotXpdr7 = 1
 {
  &EmuTeclado1 = 33
  &EmuTeclado1 = 0
 }
}

Var 1012, name XpdrInDysp, Link FSUIPC_INOUT,
Offset $0354, Length 2
{
 CALL &DisplaysSub, &XpdrInDysp
}

Var 1024, name TempXpdr, Value 1

Var 6300, name tempATC, Value 0

Var 3004, name XpdrSub, Link SUBRUTINE
{
 IF &tempATC = 0
 {
  IF &TempXpdr = 1
```

```
{
 IF &BotXpdr0 = 1
 {
  &XpdrInDysp = 0
  &TempXpdr = 2
 }
 IF &BotXpdr1 = 1
 {
  &XpdrInDysp = 4096
  &TempXpdr = 2
 }
 IF &BotXpdr2 = 1
 {
  &XpdrInDysp = 8192
  &TempXpdr = 2
 }
 IF &BotXpdr3 = 1
 {
  &XpdrInDysp = 12288
  &TempXpdr = 2
 }
 IF &BotXpdr4 = 1
 {
  &XpdrInDysp = 16384
  &TempXpdr = 2
 }
 IF &BotXpdr5 = 1
 {
  &XpdrInDysp = 20480
  &TempXpdr = 2
 }
 IF &BotXpdr6 = 1
 {
  &XpdrInDysp = 24576
  &TempXpdr = 2
 }
 IF &BotXpdr7 = 1
 {
  &XpdrInDysp = 28672
  &TempXpdr = 2
 }
}
ELSE
{
 IF &TempXpdr = 2
 {
  IF &BotXpdr0 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 0
   &TempXpdr = 3
  }
  IF &BotXpdr1 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 256
   &TempXpdr = 3
  }
  IF &BotXpdr2 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 512
   &TempXpdr = 3
  }
  IF &BotXpdr3 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 768
   &TempXpdr = 3
  }
  IF &BotXpdr4 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 1024
   &TempXpdr = 3
  }
  IF &BotXpdr5 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 1280
   &TempXpdr = 3
  }
  IF &BotXpdr6 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 1563
   &TempXpdr = 3
  }
  IF &BotXpdr7 = 1
  {
   &XpdrInDysp = &XpdrInDysp + 1792
   &TempXpdr = 3
  }
 }
 ELSE
 {
  IF &TempXpdr = 3
  {
   IF &BotXpdr0 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 0
    &TempXpdr = 4
   }
   IF &BotXpdr1 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 16
    &TempXpdr = 4
   }
   IF &BotXpdr2 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 32
    &TempXpdr = 4
   }
   IF &BotXpdr3 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 48
    &TempXpdr = 4
   }
   IF &BotXpdr4 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 64
    &TempXpdr = 4
   }
   IF &BotXpdr5 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 80
    &TempXpdr = 4
   }
   IF &BotXpdr6 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 96
    &TempXpdr = 4
   }
   IF &BotXpdr7 = 1
   {
    &XpdrInDysp = &XpdrInDysp + 112
    &TempXpdr = 4
   }
  }
  ELSE
  {
   IF &TempXpdr = 4
   {
    IF &BotXpdr0 = 1
    {
     &XpdrInDysp = &XpdrInDysp + 0
     &TempXpdr = 1
    }
    IF &BotXpdr1 = 1
    {
     &XpdrInDysp = &XpdrInDysp + 1
     &TempXpdr = 1
    }
    IF &BotXpdr2 = 1
    {
     &XpdrInDysp = &XpdrInDysp + 2
     &TempXpdr = 1
    }
    IF &BotXpdr3 = 1
```

```
        {
          &XpdrInDysp = &XpdrInDysp + 3
          &TempXpdr = 1
        }
        IF &BotXpdr4 = 1
        {
          &XpdrInDysp = &XpdrInDysp + 4
          &TempXpdr = 1
        }
        IF &BotXpdr5 = 1
        {
          &XpdrInDysp = &XpdrInDysp + 5
          &TempXpdr = 1
        }
        IF &BotXpdr6 = 1
        {
          &XpdrInDysp = &XpdrInDysp + 6
          &TempXpdr = 1
        }
        IF &BotXpdr7 = 1
        {
          &XpdrInDysp = &XpdrInDysp + 7
          &TempXpdr = 1
        }
       }
      }
     }
    }
   }
  }
}

Var 3003, name AtcSub, Link SUBRUTINE
{
 IF &BotXpdrIdent = 1
 {
  IF &BotXpdrStby = 1
  {
   &EmuTeclado1 = 19
   &EmuTeclado1 = 0
   IF &tempATC = 0
   {
    &tempATC = 1
   }
   ELSE
   {
    &tempATC = 0
   }
  }
 }
}

Var 0038, name BotXpdrIdent, Link IOCARD_SW,
Input 47, Type I
{
 CALL &AtcSub, &BotXpdrIdent
 IF &BotXpdrIdent = 1
 {
  &EmuTeclado1 = 9
  &EmuTeclado1 = 0
  &ledMiddle = 1
 }
 ELSE
 {
  &ledMiddle = 0
 }
}

Var 9500, name ContCharly, Value 0

Var 0039, name BotXpdrStby, Link IOCARD_SW,
Input 46, Type I
{
 CALL &AtcSub, &BotXpdrStby
 IF &BotXpdrStby = 1
 {
  &EmuTeclado1 = 18
  &EmuTeclado1 = 0
  IF &ContCharly = 0
  {
   &LedOuter = 1
   &ContCharly = 1
  }
  ELSE
  {
   &LedOuter = 0
   &ContCharly = 0
  }
 }
}

Var 1026, name R1R2Cambio, Link FSUIPC_INOUT,
Offset $0378, Length 2
{
 CALL &DisplaysSub
}

Var 1027, name Dme1KtRead, Link FSUIPC_INOUT,
Offset $0302, Length 2
{
 CALL &DisplaysSub
}

Var 1029, name Dme2KtRead, Link FSUIPC_INOUT,
Offset $0308, Length 2
{
 CALL &DisplaysSub
}

Var 1031, name Dme1Nm, Link FSUIPC_INOUT,
Offset $0300, Length 2
{
 CALL &DisplaysSub
}

Var 1032, name Dme2Nm, Link FSUIPC_INOUT,
Offset $0306, Length 2
{
 CALL &DisplaysSub
}

Var 0040, name IntDmeR1R2, Link IOCARD_SW,
Input 48, Type I
{
 IF &IntDmeR1R2 = 1
 {
  &R1R2Cambio = 1
 }
 ELSE
 {
  &R1R2Cambio = 2
 }
}

Var 1013, name AdfSetDysp, Link FSUIPC_INOUT,
Offset $034C, Length 2     // 2º, 3º y 4º dígitos del Adf

Var 1014, name AdfSetDyspFrac, Link
FSUIPC_INOUT, Offset $0356, Length 2     // 1º y 5º
numeros de ADF

Var 1015, name SwapComNav12, Link FSUIPC_OUT,
Offset $3123, Length 1     // Bits

Var 0042, name BotCom2Switch, Link IOCARD_SW,
Input 52, Type I
{
 IF &BotCom2Switch = 1
 {
  &SwapComNav12 = TOGGLE 2
 }
}
```

```
Var 0044, name BotNav2Switch, Link IOCARD_SW, Input 49, Type I
{
 IF &BotNav2Switch = 1
 {
  &SwapComNav12 = TOGGLE 0
 }
}

Var 0046, name BotCom1Switch, Link IOCARD_SW, Input 51, Type I
{
 IF &BotCom1Switch = 1
 {
  &SwapComNav12 = TOGGLE 3
 }
}

Var 0048, name BotNav1Switch, Link IOCARD_SW, Input 50, Type I
{
 IF &BotNav1Switch = 1
 {
  &SwapComNav12 = TOGGLE 1
 }
}

Var 0049, name EncNav1Stby, Link IOCARD_ENCODER, Input 58, Aceleration 1, Type 2

Var 1016, name AudioPanel, Link FSUIPC_INOUT, Offset $3122, Length 1
{
 CALL &AudioPanelSub, &AudioPanel
}

Var 1023, name TempAudio

Var 3002, name AudioPanelSub, Link SUBRUTINE
{
 IF &Master = 1
 {
  IF &Bateria = 1
  {
   L0 = &AudioPanel
   L1 = &AudioPanel / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudAdf = 1
   }
   ELSE
   {
    &LedAudAdf = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudDme = 1
   }
   ELSE
   {
    &LedAudDme = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudMkr = 1
   }
   ELSE
   {
    &LedAudMkr = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudNav2 = 1
   }
   ELSE
   {
    &LedAudNav2 = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudNav1 = 1
   }
   ELSE
   {
    &LedAudNav1 = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudBoth = 1
   }
   ELSE
   {
    &LedAudBoth = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudCom2 = 1
   }
   ELSE
   {
    &LedAudCom2 = 0
   }
   L0 = &TempAudio
   L1 = &TempAudio / 2
   L1 = TRUNC L1
   &TempAudio = L1
   L1 = L1 * 2
   L0 = L0 - L1
   IF L0 = 1
   {
    &LedAudCom1 = 1
   }
```

```
    ELSE
    {
     &LedAudCom1 = 0
    }
   }
   ELSE
   {
    &LedAudCom1 = 0
    &LedAudCom2 = 0
    &LedAudBoth = 0
    &LedAudNav1 = 0
    &LedAudNav2 = 0
    &LedAudMkr = 0
    &LedAudDme = 0
    &LedAudAdf = 0
   }
  }
  ELSE
  {
   &LedAudCom1 = 0
   &LedAudCom2 = 0
   &LedAudBoth = 0
   &LedAudNav1 = 0
   &LedAudNav2 = 0
   &LedAudMkr = 0
   &LedAudDme = 0
   &LedAudAdf = 0
  }
}

Var 0050, name BotAudCom1, Link IOCARD_SW,
Input 63, Type I
{
 IF &BotAudCom1 = 1
 {
  &EmuTeclado1 = 10
  &EmuTeclado1 = 0
 }
}

Var 0051, name BotAudCom2, Link IOCARD_SW,
Input 64, Type I
{
 IF &BotAudCom2 = 1
 {
  &EmuTeclado1 = 11
  &EmuTeclado1 = 0
 }
}

Var 0052, name BotAudComBoth, Link IOCARD_SW,
Input 65, Type I
{
 IF &BotAudComBoth = 1
 {
  &EmuTeclado1 = 12
  &EmuTeclado1 = 0
 }
}

Var 0053, name BotAudNav1, Link IOCARD_SW,
Input 66, Type I
{
 IF &BotAudNav1 = 1
 {
  &EmuTeclado1 = 13
  &EmuTeclado1 = 0
 }
}

Var 0054, name BotAudNav2, Link IOCARD_SW,
Input 67, Type I
{
 IF &BotAudNav2 = 1
 {
  &EmuTeclado1 = 14
  &EmuTeclado1 = 0
 }
}

Var 0055, name BotAudMkr, Link IOCARD_SW, Input
68, Type I
{
 IF &BotAudMkr = 1
 {
  &EmuTeclado1 = 15
  &EmuTeclado1 = 0
 }
}

Var 0056, name BotAudDme, Link IOCARD_SW, Input
69, Type I
{
 IF &BotAudDme = 1
 {
  &EmuTeclado1 = 16
  &EmuTeclado1 = 0
 }
}

Var 0057, name BotAudAdf, Link IOCARD_SW, Input
70, Type I
{
 IF &BotAudAdf = 1
 {
  &EmuTeclado1 = 17
  &EmuTeclado1 = 0
 }
}

Var 2500, name OuterMark, Link FSUIPC_IN, Offset
$0BB0, Length 2
{
 IF &OuterMark = 1
 {
  &LedOuter = 1
 }
 ELSE
 {
  &LedOuter = 0
 }
}

Var 2501, name MiddleMark, Link FSUIPC_IN, Offset
$0BAE, Length 2
{
 IF &MiddleMark = 1
 {
  &ledMiddle = 1
 }
 ELSE
 {
  &ledMiddle = 0
 }
}

Var 2502, name InnerMark, Link FSUIPC_IN, Offset
$0BAC, Length 2
{
 IF &InnerMark = 1
 {
  &LedInner = 1
 }
 ELSE
 {
  &LedInner = 0
 }
}

Var 2000, name LedOuter, Link IOCARD_OUT, Output
11
```

```
Var 2001, name ledMiddle, Link IOCARD_OUT, Output 12

Var 2002, name LedInner, Link IOCARD_OUT, Output 13

Var 2003, name LedAudCom1, Link IOCARD_OUT, Output 14

Var 2004, name LedAudCom2, Link IOCARD_OUT, Output 15

Var 2005, name LedAudBoth, Link IOCARD_OUT, Output 16

Var 2006, name LedAudNav1, Link IOCARD_OUT, Output 17

Var 2007, name LedAudNav2, Link IOCARD_OUT, Output 18

Var 2008, name LedAudMkr, Link IOCARD_OUT, Output 19

Var 2009, name LedAudDme, Link IOCARD_OUT, Output 20

Var 2010, name LedAudAdf, Link IOCARD_OUT, Output 21

Var 2011, name LedApAp, Link IOCARD_OUT, Output 22

Var 2012, name LedApHdg, Link IOCARD_OUT, Output 23

Var 2013, name LedApNav, Link IOCARD_OUT, Output 24

Var 2014, name LedApApr, Link IOCARD_OUT, Output 25

Var 2015, name LedApRev, Link IOCARD_OUT, Output 26

Var 2016, name LedApAlt, Link IOCARD_OUT, Output 27

Var 2017, name LedFlap0, Link IOCARD_OUT, Output 28

Var 2018, name LedFlap10, Link IOCARD_OUT, Output 29

Var 2019, name LedFlap20, Link IOCARD_OUT, Output 30

Var 2020, name LedFlapFull, Link IOCARD_OUT, Output 31

Var 4500, name Throttle, Link FSUIPC_INOUT, Offset $088C, Length 2, Type 1, Value 0

Var 4000, name PotThrottle, Link IOCARD_ANALOGIC, Input 1, PosL 0, PosC 128, PosR 255
{
 L0 = &PotThrottle - 20
 L0 = L0 * 69.71914894
 IF L0 >= 16384
 {
  &Throttle = 16384
 }
 IF L0 <= 0
 {
  &Throttle = 0
 }
 IF L0 > 0
 {
  IF L0 < 16384
  {
   &Throttle = L0
  }
 }
}

Var 4501, name Mixture, Link FSUIPC_INOUT, Offset $0890, Length 2, Type 1, Value 0

Var 4002, name PotMixture, Link IOCARD_ANALOGIC, Input 2, PosL 0, PosC 128, PosR 255
{
 &Mixture = &PotMixture * 64.25098039
}

Var 4003, name TrimVar, Link FSUIPC_INOUT, Offset $0BC0, Length 2, Type 1

Var 4004, name PotTrim, Link IOCARD_ANALOGIC, Input 3, PosL 0, PosC 128, PosR 255
{
 L0 = &PotTrim - 72
 L0 = L0 * 185.1186441
 &TrimVar = L0 - 16383
}

Var 1025, name TempApAlt

Var 1028, name TempR1Kt

Var 1030, name TempR2Kt

Var 3005, name DisplaysSub, Link SUBRUTINE
{
 IF &Master = 1
 {
  IF &Bateria = 1
  {
   &ApVsDisp = &ApVSSetDyspl
   &TempApAlt = &ApAltSetDyspl / 1997537
   &ApAltDisp = &TempApAlt * 100
   &XpdrDisp = FROMBCD &XpdrInDysp
   &AdfDisp = &ADF1Freq / 10
   &PuntoAdf = 1
   IF &R1R2Cambio = 1
   {
    &TempR1Kt = &Dme1KtRead / 10
    &DmeKtDisp = TRUNC &TempR1Kt
    &DmeNmDisp = &Dme1Nm
   }
   ELSE
   {
    &TempR2Kt = &Dme2KtRead / 10
    &DmeKtDisp = TRUNC &TempR2Kt
    &DmeNmDisp = &Dme2Nm
   }
   &O_Com1DP = 1
   &Com1SbyDisp = V8604 + 10000
   &O_COM2DP = 1
   &Com2SbyDisp = V8624 + 10000
   &oNav1Dp = 1
   &Nav1SbyDisp = V8644 + 10000
   &O_Nav2DP = 1
   &Nav2SbyDisp = V8664 + 10000
   &O_Com1DP = 1
   &Com1Disp = V8703 + 10000
   &O_Com1DP = 1
   &Com2Disp = V8713 + 10000
   &O_Com1DP = 1
   &Nav1Disp = V8723 + 10000
   &O_Com1DP = 1
```

```
    &Nav2Disp = V8733 + 10000
   }
   ELSE
   {
    &ApVsDisp = -999999
    &ApAltDisp = -999999
    &XpdrDisp = -999999
    &AdfDisp = -999999
    &PuntoAdf = 0
    &DmeKtDisp = -999999
    &DmeNmDisp = -999999
    &O_Com1DP = 0
    &Com1SbyDisp = -999999
    &O_COM2DP = 0
    &Com2SbyDisp = -999999
    &oNav1Dp = 0
    &Nav1SbyDisp = -999999
    &O_Nav2DP = 0
    &Nav2SbyDisp = -999999
    &O_Com1DP = 0
    &Com1Disp = -999999
    &O_Com1DP = 0
    &Com2Disp = -999999
    &O_Com1DP = 0
    &Nav1Disp = -999999
    &O_Com1DP = 0
    &Nav2Disp = -999999
   }
  }
  ELSE
  {
   &ApVsDisp = -999999
   &ApAltDisp = -999999
   &XpdrDisp = -999999
   &AdfDisp = -999999
   &PuntoAdf = 0
   &DmeKtDisp = -999999
   &DmeNmDisp = -999999
   &O_Com1DP = 0
   &Com1SbyDisp = -999999
   &O_COM2DP = 0
   &Com2SbyDisp = -999999
   &oNav1Dp = 0
   &Nav1SbyDisp = -999999
   &O_Nav2DP = 0
   &Nav2SbyDisp = -999999
   &O_Com1DP = 0
   &Com1Disp = -999999
   &O_Com1DP = 0
   &Com2Disp = -999999
   &O_Com1DP = 0
   &Nav1Disp = -999999
   &O_Com1DP = 0
   &Nav2Disp = -999999
  }
}

Var 5000, name ApAltDisp, Link IOCARD_DISPLAY,
Digit 5, Numbers 5     // <<<<<<<

Var 5001, name ApVsDisp, Link IOCARD_DISPLAY,
Digit 0, Numbers 5     // <<<<<<<

Var 5002, name XpdrDisp, Link IOCARD_DISPLAY,
Digit 10, Numbers 4     // <<<<<<<<

Var 6700, name X_ADF1High, Link FSUIPC_INOUT,
Offset $034C, Length 2
{
  L0 = FROMBCD &X_ADF1High
  IF &ADF1High <> L0      // block values coming from
rotary
  {
    &ADF1High = L0
    CALL &CalcADF1Freq
  }
}

Var 6701, name X_ADF1Low, Link FSUIPC_INOUT,
Offset $0356, Length 2
{
  IF &X_ADF1Low <> &ADF1Low       // block values
coming from rotary
  {
    &ADF1Low = &X_ADF1Low
    CALL &CalcADF1Freq
  }
}

Var 6710, name ADF1High       // FSUIPC ADF value
high: 3 digits

Var 6711, name ADF1Low      // FSUIPC ADF value: 1
digit

Var 6712, name ADF1Freq        // total (needed for
display): 4 digits

Var 6713, name CalcADF1Freq, Link SUBRUTINE
{
  L0 = &ADF1High * 10     // high * 10
  &ADF1Freq = L0 + &ADF1Low     // + decimal
  CALL &DisplaysSub
}

Var 6744, name RO_ADF1High, Link
IOCARD_ENCODER, Input 54, Aceleration 4, Type 2
// Poner salida que sea
{
  L0 = &RO_ADF1High * -1     // change direction
  &ADF1High = ROTATE 200 ,990 ,L0
  CALL &CalcADF1Freq
  &X_ADF1High = TOBCD &ADF1High       // higher 3
Digits in bcd to fsuipc
}

Var 0058, name RO_ADF1Low, Link
IOCARD_ENCODER, Input 99, Aceleration 1, Type 2
// No se usa, solo si tenemos 2 enco
{
  L0 = &RO_ADF1Low * 5     // in steps of 5
  &ADF1Low = ROTATE 0 ,9 ,L0
  CALL &CalcADF1Freq
  &X_ADF1Low = &ADF1Low          //  decimal digit to
fsuipc
}

Var 5003, name AdfDisp, Link IOCARD_DISPLAY,
Digit 16, Numbers 3     // <<<<<<<<

Var 2021, name PuntoAdf, Link IOCARD_OUT, Output
32

Var 5004, name DmeKtDisp, Link IOCARD_DISPLAY,
Digit 20, Numbers 3     // <<<<<<<<

Var 5005, name DmeNmDisp, Link
IOCARD_DISPLAY, Digit 32, Numbers 3          //
<<<<<<<<<

Var 8600, name SbyCom1Freq, Link FSUIPC_IN,
Offset $311A, Length 2     // Standby COM1 Freq
{
  L0 = FROMBCD &SbyCom1Freq
  V8604 = L0
  V8603 = DIV L0 ,100
  V8602 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8602     // COM1FreqLow
```

```
Var 8603     // COM1FreqHigh

Var 8604     // COM1Freq

Var 8605, name CalcCOm1Fr, Link SUBRUTINE     //
CalcCOM1Freq
{
  L0 = V8603 * 100     // high * 100
  V8604 = L0 + V8602     // + low
  &Com1SbyFreq = TOBCD V8604     // BCD value to
panel
  CALL &DisplaysSub     // display new freq value
}

Var 8609, name Com1Mode, Link IOCARD_SW, Input
73, Type P     // Com1Mode push button

Var 8610, name RO_Com1, Link
IOCARD_ENCODER, Input 24, Aceleration 1, Type 2
// Com1
{
  L0 = &RO_Com1
  IF &Com1Mode = 1
  {
    V8603 = ROTATE 18 ,36 ,L0
  }
  ELSE
  {
    L0 = L0 * 5
    V8602 = ROTATE 0 ,99 ,L0
  }
  CALL &CalcCOm1Fr
}

Var 8614, name O_Com1DP, Link IOCARD_OUT,
Output 99     // O_ComDP

Var 8616, name Com1SbyFreq, Link FSUIPC_OUT,
Offset $311A, Length 2     // Standby COM1 Freq

Var 8620, name Com2SbyFreq, Link FSUIPC_IN,
Offset $311C, Length 2     // Standby COM2 Freq
{
  L0 = FROMBCD &Com2SbyFreq
  V8624 = L0
  V8623 = DIV L0 ,100
  V8622 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8622     // COM2FreqLow

Var 8623     // COM2FreqHigh

Var 8624     // COM2Freq

Var 8625, name CalcCOm2Freq, Link SUBRUTINE
// CalcCOM2Freq
{
  L0 = V8623 * 100     // high * 100
  V8624 = L0 + V8622     // + low
  V8636 = TOBCD V8624     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8629, name Com2Push, Link IOCARD_SW, Input
72, Type P     // COM2 Mode push button

Var 8630, name Com2FLRot, Link
IOCARD_ENCODER, Input 60, Aceleration 1, Type 2
// RO_NAV1FL
{
  L0 = &Com2FLRot
  IF &Com2Push = 1
  {
    V8623 = ROTATE 18 ,36 ,L0
  }
  ELSE
  {
    L0 = L0 * 5
    V8622 = ROTATE 0 ,99 ,L0
  }
  CALL &CalcCOm2Freq
}

Var 8634, name O_COM2DP, Link IOCARD_OUT,
Output 99     // O_COM2DP

Var 8635, Link FSUIPC_INOUT, Offset $0C4E, Length
2     // FO_NAV1Crs

Var 8636, Link FSUIPC_OUT, Offset $311C, Length 2
// Standby COM2 Freq

Var 8640, Link FSUIPC_IN, Offset $311E, Length 2
// FI_NAV1Freq
{
  L0 = FROMBCD V8640
  V8644 = L0
  V8643 = DIV L0 ,100
  V8642 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8642     // NAV1FreqLow

Var 8643     // NAV1FreqHigh

Var 8644     // NAV1Freq

Var 8645, name CalcNav1Freq, Link SUBRUTINE     //
CalcNAV1Freq
{
  L0 = V8643 * 100     // high * 100
  V8644 = L0 + V8642     // + low
  &FONAV1Freq = TOBCD V8644     // BCD value to
panel
  CALL &DisplaysSub     // display new freq value
}

Var 8649, name Nav1Push, Link IOCARD_SW, Input
74, Type P     // NAV1Mode push button

Var 8650, name Nav1Rot, Link IOCARD_ENCODER,
Input 58, Aceleration 1, Type 2     // RO_NAV1FL
{
  L0 = &Nav1Rot
  IF &Nav1Push = 1
  {
    V8643 = ROTATE 8 ,17 ,L0
  }
  ELSE
  {
    L0 = L0 * 5
    V8642 = ROTATE 0 ,99 ,L0
  }
  CALL &CalcNav1Freq
}

Var 8654, name oNav1Dp, Link IOCARD_OUT, Output
99     // O_NAV1DP

Var 8656, name FONAV1Freq, Link FSUIPC_OUT,
Offset $311E, Length 2     // FO_NAV1Freq

Var 8660, name Nav2SbyFreq, Link FSUIPC_IN,
Offset $3120, Length 2     // NAV2 Standby Freq
{
  L0 = FROMBCD &Nav2SbyFreq
  V8664 = L0
  V8663 = DIV L0 ,100
  V8662 = MOD L0 ,100
```

```
  CALL &DisplaysSub     // display new freq value
}

Var 8662     // NAV2FreqLow

Var 8663     // NAV1FreqHigh

Var 8664     // NAV2Freq

Var 8665, name CalcNav2, Link SUBRUTINE      //
CalcNAV2Freq
{
  L0 = V8663 * 100     // high * 100
  V8664 = L0 + V8662     // + low
  V8676 = TOBCD V8664     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8669, name Nav2Push, Link IOCARD_SW, Input
75, Type P     // NAV2Mode push button

Var      8670,      name      nav2FLROT,      Link
IOCARD_ENCODER, Input 56, Aceleration 1, Type 2
// RO_NAV2FL
{
  L0 = &nav2FLROT
  IF &Nav2Push = 1
  {
    V8663 = ROTATE 8 ,17 ,L0
  }
  ELSE
  {
    L0 = L0 * 5
    V8662 = ROTATE 0 ,99 ,L0
  }
  CALL &CalcNav2
}

Var 8674, name O_Nav2DP, Link IOCARD_OUT,
Output 96     // O_NAV2DP

Var 8675, Link FSUIPC_INOUT, Offset $0C4E, Length
2     // FO_NAV2Crs

Var 8676, Link FSUIPC_OUT, Offset $3120, Length 2
// NAV2 Standby Freq

Var 8700, Link FSUIPC_IN, Offset $034E, Length 2
// Active COM1 Freq
{
  L0 = FROMBCD V8700
  V8703 = L0
  V8702 = DIV L0 ,100
  V8701 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8701     // COM1FreqLow

Var 8702     // COM1FreqHigh

Var 8703     // COM1Freq

Var 8704, name CalcCom1Frr, Link SUBRUTINE      //
CalcCOM1Freq
{
  L0 = V8702 * 100     // high * 100
  V8703 = L0 + V8701     // + low
  V8707 = TOBCD V8703     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8707, Link FSUIPC_OUT, Offset $034E, Length 2
// Active COM1 Freq

Var 8710, Link FSUIPC_IN, Offset $3118, Length 2
// Active COM2 Freq
{
  L0 = FROMBCD V8710
  V8713 = L0
  V8712 = DIV L0 ,100
  V8711 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8711     // COM1FreqLow

Var 8712     // COM1FreqHigh

Var 8713     // COM1Freq

Var 8714, name CalcCOm1Fre, Link SUBRUTINE      //
CalcCOM1Freq
{
  L0 = V8712 * 100     // high * 100
  V8713 = L0 + V8711     // + low
  V8717 = TOBCD V8713     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8717, Link FSUIPC_OUT, Offset $3118, Length 2
// Active COM2 Freq

Var 8720, Link FSUIPC_IN, Offset $0350, Length 2
// Active NAV1 Freq
{
  L0 = FROMBCD V8720
  V8723 = L0
  V8722 = DIV L0 ,100
  V8721 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8721     // NAV1FreqLow

Var 8722     // NAV1FreqHigh

Var 8723     // NAV1Freq

Var 8724, name CalcNav1Fre, Link SUBRUTINE      //
CalcNAV1Freq
{
  L0 = V8722 * 100     // high * 100
  V8723 = L0 + V8721     // + low
  V8727 = TOBCD V8723     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8727, Link FSUIPC_OUT, Offset $0350, Length 2
// Active NAV1 Freq

Var 8730, Link FSUIPC_IN, Offset $0352, Length 2
// Active NAV1 Freq
{
  L0 = FROMBCD V8730
  V8733 = L0
  V8732 = DIV L0 ,100
  V8731 = MOD L0 ,100
  CALL &DisplaysSub     // display new freq value
}

Var 8731     // NAV2FreqLow

Var 8732     // NAV2FreqHigh

Var 8733     // NAV2Freq

Var 8734, name CalcNav2Fr, Link SUBRUTINE      //
CalcNAV2Freq
{
  L0 = V8732 * 100     // high * 100
```

```
  V8733 = L0 + V8731     // + low
  V8737 = TOBCD V8733     // BCD value to panel
  CALL &DisplaysSub     // display new freq value
}

Var 8737, Link FSUIPC_OUT, Offset $0352, Length 2
// Active NAV2 Freq

Var 8800, name Com1Disp, Link IOCARD_DISPLAY,
Digit 40, Numbers 5     // Active COM1

Var 8801, name Com1SbyDisp, Link
IOCARD_DISPLAY, Digit 64, Numbers 5     // Standby
COM1

Var 8802, name Com2Disp, Link IOCARD_DISPLAY,
Digit 35, Numbers 5     // Active COM2

Var 8803, name Com2SbyDisp, Link
IOCARD_DISPLAY, Digit 48, Numbers 5     // Standby
COM2

Var 8804, name Nav1Disp, Link IOCARD_DISPLAY,
Digit 69, Numbers 5     // Active NAV 1

Var 8805, name Nav1SbyDisp, Link
IOCARD_DISPLAY, Digit 74, Numbers 5     // Standby
NAV 1

Var 8806, name Nav2Disp, Link IOCARD_DISPLAY,
Digit 53, Numbers 5     // Active NAV 2

Var 8807, name Nav2SbyDisp, Link
IOCARD_DISPLAY, Digit 58, Numbers 5     // Standby
NAV 2

Var 8809, name AdfPush, Link IOCARD_SW, Input 76
{
  IF &AdfPush = 1
  {
    &EmuTeclado1 = 26
    &EmuTeclado1 = 0
  }
}

Var 2200, name TestPitch

Var 2201, name TestBank

Var 2203, Value 0
{
  &PitchServo = 512
  &BankServo = 512
}

Var 2204, name PitchServo, Link USB_SERVOS,
Output 1, PosL 0, PosC 512, PosR 1023

Var 2205, name BankServo, Link USB_SERVOS,
Output 3, PosL 0, PosC 512, PosR 1023

Var 2206, name Pitch, Link FSUIPC_IN, Offset $0578,
Length 4
{
  IF &Master = 1
  {
    L0 = &Pitch * 8.38E-008
    L0 = L0 + 10
    &TestPitch = L0
    IF L0 > 20
    {
      L0 = 20
    }
    IF L0 < 0
    {
      L0 = 0
    }
    &PitchServo = L0 * 51.15
  }
  ELSE
  {
    &PitchServo = 512
  }
}

Var 2207, name Bank, Link FSUIPC_IN, Offset $057C,
Length 4
{
  IF &Master = 1
  {
    L0 = &Bank * 8.38E-008
    L0 = L0 + 10
    &TestBank = L0
    IF L0 > 20
    {
      L0 = 20
    }
    IF L0 < 0
    {
      L0 = 0
    }
    &BankServo = L0 * 51.15
  }
  ELSE
  {
    &BankServo = 512
  }
}

Var 1112, name Kollsman, Link FSUIPC_OUT, Offset
$0330, Length 2, Value 16208

Var 1113, name HDG, Link FSUIPC_OUT, Offset
$07CC, Length 2, Value 0

Var 1114, name Push, Link FSUIPC_OUT, Offset
$0C3E, Length 2, Value 1

Var 1115, name OBS1, Link FSUIPC_OUT, Offset
$0C4E, Length 2, Value 1

Var 1116, name OBS2, Link FSUIPC_OUT, Offset
$0C5E, Length 2, Value 1

Var 1117, name Adf_Rotat, Link FSUIPC_OUT, Offset
$0C6C, Length 2, Value 1


Var 0115, name EncKollsman, Link
IOCARD_ENCODER, Input 18, Aceleration 1, Type 2
{
  L0 = &EncKollsman * 16
  &Kollsman = &Kollsman - L0
}

Var 3100, name HdgTemp

Var 0117, name EncHDG, Link IOCARD_ENCODER,
Input 24, Aceleration 1, Type 2
{
  &HdgTemp = ROTATE 0 ,359 ,&EncHDG
  &HDG = &HdgTemp * 182.0444444
}

Var 3101, name PushTemp

Var 0118, name EncPush, Link IOCARD_ENCODER,
Input 22, Aceleration 1, Type 2
{
  &PushTemp = ROTATE 0 ,359 ,&EncPush
  &Push = &PushTemp * 182.0444444
}
```

```
Var 0119, name EncObs1, Link IOCARD_ENCODER,
Input 20, Aceleration 1, Type 2
{
  &OBS1 = ROTATE 0 ,359 ,&EncObs1
}

Var 0120, name EncObs2, Link IOCARD_ENCODER,
Input 27, Aceleration 1, Type 2
{
  &OBS2 = ROTATE 0 ,359 ,&EncObs2
}

Var 0121, name EncAdf, Link IOCARD_ENCODER,
Input 29, Aceleration 1, Type 2
{
  &Adf_Rotat = ROTATE 0 ,359 ,&EncAdf
}
```

2. Configuración de “sioc.ini”

```
[ fichero de configuracion para el SIOC ver. 4.5 ]
[ Configuration file for SIOC ]

[********** SIOC **********]

[ Nombre asignado al SIOC ]
[ SIOC name ]
Name=Main

[ Puerto del servidor IOCP ]
[ IOCP port ]
IOCP_port=8092

[ Tiempo de respuesta máximo de los paquetes IOCP ]
[ IOCP Timeout ]
IOCP_timeout=4000

[ Arranque minimizado en la barra ]
[ Start minimized in tray ]
Minimized=No

[ Retraso necesario para las variables toggles (Project Magenta)]
[ Deley needed for var. toggles (Project Magenta) ]
toggle_delay=20

[ Fichero de configuracion ]
[ Configuration File ]
CONFIG_FILE=.\portatil.ssi

[********** MONITOR MODULE **********]

[ Desabilitar monitorizacion remota del SIOC ]
[ Remote monitor disable mode yes/no ]
Monitor_disable=No

[ Refresco broadcast monitor (en segundos) ]
[ Broadcast monitor time (seconds) ]
Monitor_time=2


[********** IOCARDS MODULE **********]

[ Desactivar el módulo de las IOCards ]
[ Disable IOCards module ]
IOCard_disable=No

[ Divisor de frecuencia para los ejes analogicos (1-999). Retarda la entrega de valores. ]
[ Frequency divisor for delay the analogic axes. (1-999)]
Divisor_AD=10

[---------- CARDS CONFIG ----------]

[ IOCard Master ]
[.....]

[ Spanish : ]

[ MASTER=(Indice device),(Tipo),(Número de tarjetas),(Número device) ]

[ Indice Device: Índice usado en la variable SIOC como device, para indicar a que tarjeta se hace referencia ]
[ Se usa 0 en el caso de que sólo haya una tarjeta y no se tenga en cuenta el número de device, por defecto un script en SIOC cuando no se pone parámetro DEVICE, se hace referencia al índice 0]

[ tipo = 0 : Emulador de Master Card // OBSOLETO ]
[ tipo = 1 : Tarjeta Master conectada al puerto paralelo directamente // OBSOLETO ]
[ tipo = 2 : Tarjeta Master conectada al puerto paralelo mediante cable de compatibilidad // OBSOLETO ]
[ tipo = 3 : Placa de expansión por puerto paralelo // OBSOLETO ]
[ tipo = 4 : Tarjeta USBExpansion usada ]
[ tipo = 5 : Modulo MCP de Opencockpits ]
[ tipo = 6 : Tarjeta USBOutputs ]
[ tipo = 7 : Modulo EFIS de Opencockpits ]
[ tipo = 8 : Modulo Radio COM de Opencockpits ]
[ tipo = 9 : Modulo Radio NAV de Opencockpits ]
[ tipo = 10 : Modulo Radio ADF de Opencockpits ]
[ tipo = 11 : Modulo Radio ATC de Opencockpits ]
[ tipo = 12 : Modulo Radio Airbus RMP de Opencockpits ]
[ tipo = 13 : Modulo FMC-737 de Opencockpits ]
[ tipo = 14 : Tarjeta USBDcmotorPLUS ]
[ tipo = 15 : Modulo MCP V3 de Opencockpits ]
[ tipo = 16 : Modulo CRONOMETRO B737 ]


[ Número de tarjetas = Número de placas Master usadas, 1 a 4 para uso de placas de expansión, 1 para conexión de placa Master directa o del Emulador ]

[ Número de device = 0 en el caso del Emulador o primera tarjeta USB detectada, Dirección del puerto paralelo (por ejemplo $0378), o número de dispositivo USB ]


[ Un ejemplo de 2 USBExpansion conectadas con 3 y 2 placas Master ]
[ MASTER=0,4,3,22 ]
[ MASTER=1,4,2,24 ]

[ Un ejemplo de conexión a simulador ]
[ MASTER=0,0,1,0 ]


[ English : ]

[ MASTER=(Device index),(Type),(Number of cards),(Device number) ]

[ Device index : Index used in SIOC variable like device, this is the card used for SIOC sentence ]
[ Use 0 for only one card, for this value you not need specify Device number. If you not use DEVICE parameter in a definition in SIOC script, the default index used is 0]

[ type = 0 : Master Card Emulator // OBSOLETE ]
[ type = 1 : Master Card connected directly to parallel port // OBSOLETE ]
[ type = 2 : Master Card connected throught compatibility cable to parallel port //OBSOLETE]
[ type = 3 : Expansion Card connected throught parallel port //OBSOLETE ]
[ type = 4 : USBExpansion Card used ]
```

[type = 5 : Opencockpits MCP module]
[type = 6 : USBOutputs Card used]
[type = 7 : EFIS module]
[type = 8 : Radio COM module]
[type = 9 : Radio NAV module]
[type = 10 : Radio ADF module]
[type = 11 : Radio ATC module]
[type = 12 : Radio RMP Airbus module]
[type = 13 : FMC-737 module]
[type = 14 : USBDCmotorPLUS Card used]
[type = 15 : MCP V3 module]
[type = 16 : CHRONO B737 module]

[Number of Cards = Master cards connecteds, 1 to 4 for expansion cards, 1 for a Master card directly connected or Emulator]

[Number of device = 0 for Emulator or first USB card detected, parallel port address, device number for specifies USBexpansion card]

[For example, two USBExpansion cards connected with 3 and 2 Master cards used]
[MASTER=0,4,3,22]
[MASTER=1,4,2,24]

[For example, use of Master card emulator]
[MASTER=0,0,1,0]

[Use the first USBExpansion card connected with only one Master Card attached]
[Esta definición es para usar la primera tarjeta USBExpansion que se encuentre instalada que además llevará conectada una placa Master]

[MASTER=3,6,1,252]
MASTER=0,4,2,0
[MASTER=0,12,1,0]
[MASTER=0,16,1,0]

[Others Cards / Otras tarjetas]
[.....]

[Spanish :]

[Nombre_de_tarjeta=(Indice device),(Número device)]

[Indice Device: Índice usado en la variable SIOC como device, para indicar a que tarjeta se hace referencia]
[Se usa 0 en el caso de que sólo haya una tarjeta y no se tenga en cuenta el número de device, por defecto un script en SIOC cuando no se pone parámetro DEVICE, se hace referencia al índice 0]

[Número de device = 0 para usar primera tarjeta USB de este tipo detectada ó número de dispositivo USB]

[English :]

[Name_of_card=(Device index),(Device number)]

[Device index : Index used in SIOC variable like device, this is the card used for SIOC sentence]
[Use 0 for only one card, for this value you not need specify Device number. If you not use DEVICE parameter in a definition in SIOC script, the default index used is 0]

[Number of device = 0 for first USB card of this type detected, or device number for a specifies device]

[Ejemplo de dos USBServos en indices 0 y 1, y números de devices 17 y 23]

[Example of two USBServos for index 0 and 1, and device number 17 and 23]

[USBServos=0,17]
[USBServos=1,23]

USBStepper=0,0
USBKeys=0,0
USBServos=0,0
USBRelays=0,0
USBDCmotor=0,0

[Para los ejes analógicos se usan los números de devices de las tarjetas donde están alojados]
[For analogic axles, you use the device number of cards what allow the axles]

USBAnalogic=0,0

[********** FSUIPC MODULE **********]

[Desabilitar lectura de las FSUIPC]
[FSUIPC disable mode yes/no]
FSUipcdisable=No

[Refresco recepción FSUIPC]
[FSUIPC refresh]
FSUipcRefresh=50

[********** IOCP CLIENTS MODULES **********]

[Retraso para inicializacion una vez conectado el cliente en milisegundos]
[Delay for initialization when client has been connected in mseconds]
IOCPini_delay=3000

[********** IOCP CLIENT MODULE #0 **********]

[Desactivar el módulo cliente IOCP]
[Disable IOCP client module]
IOCPclient0_disable=No

[IP del servidor donde debe de conectar el cliente]
[IOCP client host name]
IOCPclient0_host=localhost

[Puerto de envio del protocolo IOCP cliente]
[IOCP client port]
IOCPclient0_port=8099

[********** IOCP CLIENT MODULE #1 **********]

[Desactivar el módulo cliente IOCP]
[Disable IOCP client module]
IOCPclient1_disable=Yes

[IP del servidor donde debe de conectar el cliente]
[IOCP client host name]
IOCPclient1_host=localhost

[Puerto de envio del protocolo IOCP cliente]
[IOCP client port]
IOCPclient1_port=8099

[********** SOUND MODULE **********]

[Ficheros de sonido]

[Sound Files]

[Desactivar el módulo de sonido]
[Disable Sound module]
Sound_disable=yes

[Volumen general de los sonidos 0-100]
[Master Volume 0-100]
Volume=100

[put '*' first filename for loop Sound]
[anteponer '*' en el fichero para bucle continuo de sonido]

[Sound=wav_file,frequency,volume,pan]
[frequency=100 to 100000 0=original -1=current]
[volume=0 to 100, -1=current]
[pan=-100 (left) to +100 (right) 0=center -1=current]

[Sound=Fichero_wav,frecuencia,volumen,balance]
[frecuencia=100 hasta 100000 0=original -1=Por defecto]
[volumen=0 hasta 100 -1=Volumen por defecto]
[balance=-100 (Izquierda) hasta +100 (Derecha) 0=centro -1=Por defecto]

[#1]
Sound=APDis.wav,-1,-1,-1

[#2]
Sound=*outermk.wav,-1,-1,-1

[#3]
Sound=*hello.wav

[********** KEYBOARD EMULATOR MODULE **********]

[Nombre exacto de la ventana donde se enviarán las teclas]
[Name of window for key send]
[window = "Project Magenta Glass Cockpit - Build 396"]

window ="Microsoft Flight Simulator X"

[Asignación de teclas]
[assign youe keys]

#1=A
#2=B
#3=C
#4=D
#5=E
#6=F
#7=G
#8=H
#9=\3I\4
#10=J
#11=K
#12=L
#13=M
#14=N
#15=O
#16=\34\4
#17=Q
#18=\3R\4
#19=I
#20=T
#21=U
#22=V
#23=W
#24=X
#25=Y
#26=Z
#27=1
#28=2
#29=3
#30=4
#31=5
#32=6
#33=7
#34=8
#35=9
#36=0
#37=<
#38=,
#39=.
#40=-
#41=*
#42=+
#43=//
#44=/0
#45=/1
#46=/A
#47=
#48=
#49=
#50=
#51=
#52=
#53=
#54=
#55=
#56=
#57=
#58=
#59=
#60=
#61=
#62=
#63=
#64=
#65=
#66=
#67=
#68=
#69=
#70=
#71=
#72=
#73=
#74=
#75=
#76=
#77=
#78=
#79=
#80=
#81=
#82=
#83=
#84=
#85=
#86=
#87=
#88=<a
#200=B